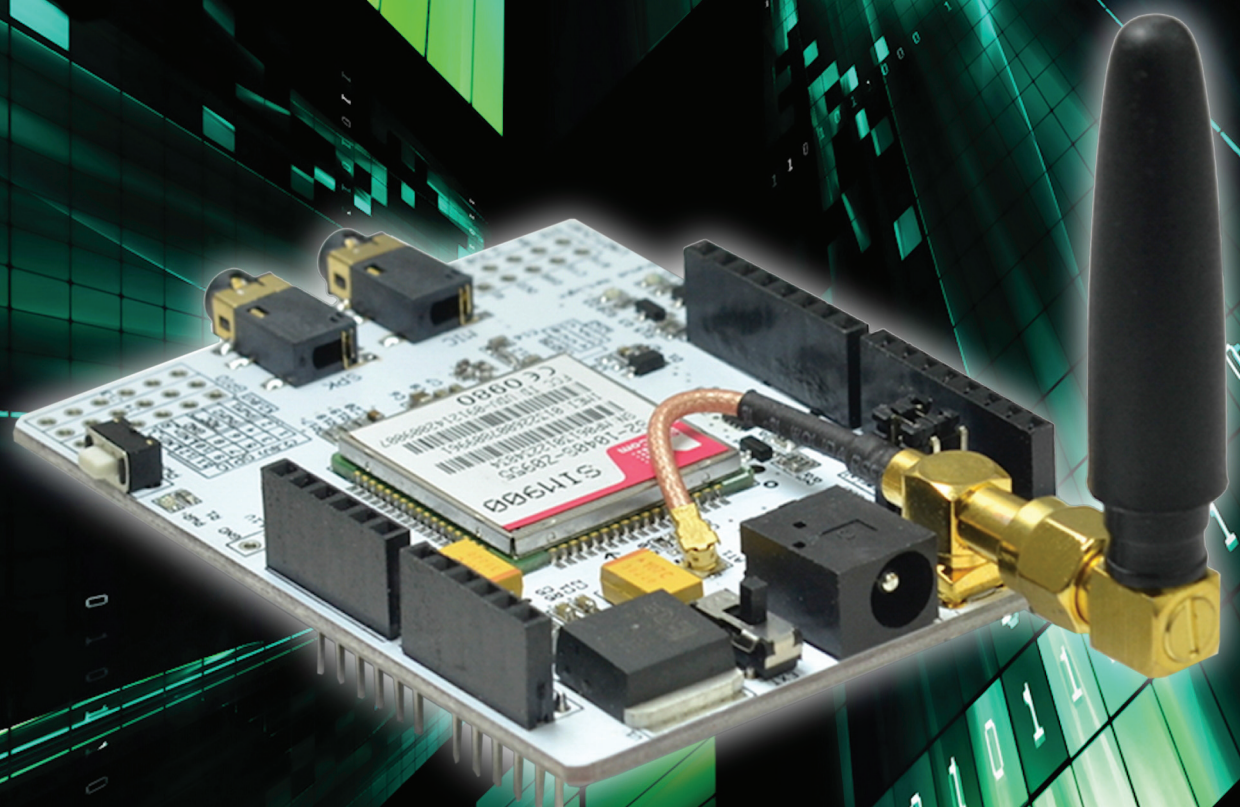
fare elettronica
n.347 - Settembre 2014

monitoraggio con ARDUINO

stazione di controllo GSM-GPRS



FPGA & ASIC

Un tutorial per impararne la programmazione

Usare mikroC per la gestione delle uscite
Sensori di pressione e di umidità
Controllo accessi con PIC

Fare Elettronica n.347 – Settembre 2014

Sommario

l'elettronica è qui.

Il nuovo spazio dedicato ai progettisti elettronici e ai makers

IE cloud
www.ie-cloud.it

INWARE EDIZIONI

Il nuovo portale IEcloud mette a disposizione degli utenti numerosi ed interessanti contenuti in tema di elettronica.
Progetti, articoli e news possono essere condivisi nella community e fruiti in tempo reale da tutti i membri.

IEcloud è il portale di riferimento per tutti i professionisti, progettisti, studenti e appassionati di elettronica.

Registrati subito, è GRATIS!

- Centinaia di articoli, riviste, ebook, video, pdf sempre a tua disposizione
- Una community per condividere i propri progetti o per cercare collaborazioni
- Notizie, aggiornamenti ed eventi relativi al mondo dell'elettronica
- Un portale fruibile da qualsiasi dispositivo smartphone, tablet o PC

	FREE	SMART	MAKER	GENIUS
	GRATIS per sempre	€5.99/mese o 59,99/anno	€5.99/mese o 59,99/anno	€7.99/mese o 79,99/anno
Accesso a news ed eventi	✓	✓	✓	✓
Accesso alla community	✓	✓	✓	✓
Accesso ai progetti gratuiti	✓	✓	✓	✓
Accesso ai progetti premium	✗	✓	✓	✓
Accesso a tutte le riviste Fare Elettronica	✗	✗	✓	✓
Accesso a tutte le riviste Firmware	✗	✓	✗	✓
	Registrati	Acquista	Acquista	Acquista

STAZIONE DI CONTROLLO GPRS

di Angelo De Bartolo

Una stazione di controllo GPRS che monitora costantemente degli ingressi è una soluzione più sicura di un combinatore telefonico GSM. Con un budget limitato è possibile costruire un sistema completo, dalla stazione periferica fino al server centrale.

Ci si potrebbe chiedere il motivo di utilizzare una connessione GPRS come sistema di segnalazione di uno stato d'allarme quando in commercio esistono numerosi combinatori telefonici GSM ad ottimi prezzi. Un motivo risiede nella sicurezza: i combinatori GSM effettuano una chiamata vocale nel momento in cui scatta l'ingresso d'allarme pertanto basterebbe utilizzare un disturbatore di frequenze (uno jammer GSM, facilmente reperibile in rete) per eludere tale controllo. La nostra stazione, invece, si collegherà tramite GPRS ad un server (che realizzeremo a partire da zero) ad intervalli regolari: se la stazione periferica non si collega entro un tempo massimo di timeout si avviserà della mancata connessione.

Figura 1: Il sistema da realizzare

Scelta del modulo GPRS

Il modulo di comunicazione scelto è il famosissimo SIM900 della SIMCom, un modulo quad-band GSM-GPRS economico, molto diffuso ed utilizzato in diverse board. Per una maggiore semplicità di utilizzo della SIM900 si è preferito acquistare il chip già montato su una board completa di tutto l'occorrente (jack microfono, jack cuffie, alimentazione, ecc. ecc.) in formato compatibile con le shield Arduino: utilizzeremo lo shield EFCOM PRO GPRS/GSM MODULE prodotta da ElecFreaks Studio (figura 1), acquistabile a meno di 50€ in rete. Il modulo è un Quad-Band 850 / 900/ 1800/ 1900 MHz multislot classe 10/8. La descrizione dei pin è rappresentata nella tabella 1. La comunicazione avviene tramite due pin utilizzando una trasmissione seriale UART emulata tramite software oppure se utilizziamo una eventuale UART hardware presente. L'alimentazione deve essere in grado di fornire 2A poiché in alcuni momenti il modulo assorbe molta energia e quindi bisogna assolutamente evitare di collegare la scheda ai +5V di Arduino.

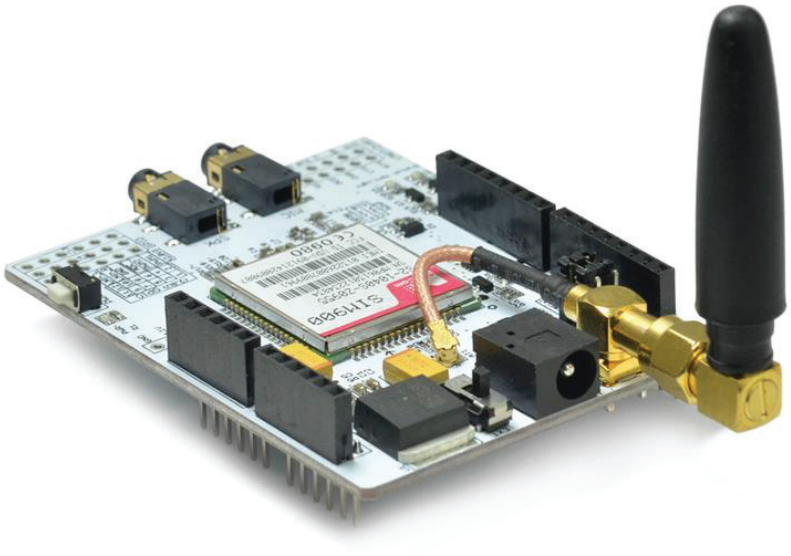


Figura 2: Lo shield GSM/GPRS della EFCOM

Primo avvio della scheda.

Prima di avviare la scheda è necessario assicurarsi di avere un'alimentazione esterna capace di erogare almeno 2A (come si legge dal datasheet del modulo SIM900) dato che durante le trasmissioni GSM/GPRS ci possono essere dei picchi di assorbimento pari alla portata indicata: l'alimentazione esterna la si può applicare tramite il jack presente ricordando che è ammessa una tensione compresa tra 4,8VDC÷5VDC. Eventualmente per trasmissioni brevi si potrebbe provare ad utilizzare l'alimentazione della board di Arduino (assumendosi il rischio, però, di poter danneggiare Arduino): a tale scopo è stato inserito un jumper per selezionale l'alimentazione dalla board Arduino o dal jack esterno. Bisogna poi inserire la SIM CARD nello slot ed assicurarsi di averla bloccata in posizione LOCK; al fine di un corretto funzionamento la SIM non deve avere alcun codice PIN attivo. Ora basta solo innestare collegare la shield nell'Arduino e collegare Arduino al PC tramite USB e la shield sarà alimentata (a conferma di ciò basta verificare che lo STATUS-LED di colore verde si accenda). Per accendere il dispositivo sono presenti un PWR-KEY (accensione hardware) ed un pin (D9 per l'accensione via software) dedicati all'accensione e spegnimento del dispositivo: una volta acceso (via hardware o software) il led NET-LIGHT (di colore rosso) del dispositivo inizierà a lampeggiare con frequenza di 1 ogni 800ms circa fino all'aggancio della rete: una volta che la SIM si sarà registrata sulla rete del nostro operatore telefonico i lampeggi saranno più lenti (alla frequenza di 1 ogni 3 secondi circa).

Creazione del server web

Un server web per "uso domestico" può essere facilmente configurabile installando il software XAMPP sia su piattaforme Windows che Linux o anche Mac. Inutile dilungarsi sull'installazione ed altri approfondimenti dato che in rete si trovano guide, tutorial e documentazione varia anche in italiano. Ancor meglio sarebbe virtualizzare il server in modo tale che se eventualmente venisse infettato o attaccato dall'esterno il tutto coinvolgerebbe solo il sistema virtuale. Per questa prova, invece, ho preferito utilizzare come server non un PC personale configurato ad-hoc ma un server esterno e la scelta è

ricaduta in uno dei più famosi web hosting: altervista.
La registrazione di uno spazio gratuito è semplicissima, basta inserire il nome del sito che si vuole creare (in questo caso si è scelto http://sicur.altervista.it) ed inserire i propri dati personali. Per la tipologia di utilizzo adatta ai nostri scopi non è necessario creare alcun contenuto del sito e pertanto si utilizzerà esclusivamente il pannello di controllo per caricare i file php necessari al funzionamento del server.
La funzione che il nostro server dovrà svolgere è la "custodia" dei valori degli ingressi della stazione GPRS selezionata: i dati vengono salvati su file TXT residenti nel server il cui nome del file inizia con il numero di telefono della stazione seguito da un simbolo di underline e le diciture "IN" oppure "PWD" a seconda se dobbiamo lavorare con gli ingressi (IN) o con la password (PWD). Supponendo come numero di telefono della stazione 3331234567 si adopereranno due files:
-un file "3331234567_IN.txt" che contiene al suo interno il valore degli ingressi;
-un file "3331234567_PWD.txt" che contiene al suo interno la password;

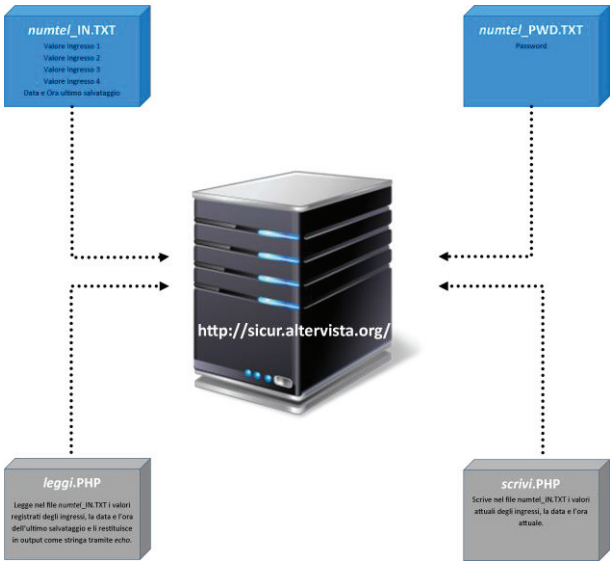


Figura 3: I file di dati (in blu) e delle funzioni (in grigio) contenuti nel Server

Il codice è abbastanza semplice ed intuitivo anche per chi (come me) non è un programmatore esperto del PHP: i valori delle variabili vengono passate direttamente nella url del sito (tipico passaggio dei parametri con metodo GET) mentre per essere recuperate viene utilizzato all'interno dello script PHP il metodo GET; una volta recuperate le variabili si verifica che la password passata nella url sia coincidente con quella indicata nel file "numtel_PWD.txt" e solo in questo caso si procede a leggere o scrivere i dati nei files.

Leggere e scrivere i valori degli ingressi

La board Arduino aggiornerà il file "numtel_IN.txt" ogni qualvolta che almeno uno degli ingressi cambia stato; per fare ciò Arduino si connetterà ad internet richiamando l'URL dello script scrivi.php passando come parametri il numero di telefono, i valori dei singoli ingressi e la password. Lo script apre il file in modalità "w+" (modalità scrittura partendo dall'inizio del file) ovvero se il file non esiste viene creato mentre se già esiste viene sovrascritto e la scrittura parte dall'inizio del file, in coda al fine viene invece inserita la data e l'ora corrente formattate secondo le proprie preferenze. Un contenuto tipico del file "numtel_IN.txt" può essere il seguente:

$1%1%1%0%0015%01/09/2014 12:00:00

indicante che gli ingressi 1, 2 e 3 sono al livello logico High (valore 1) mentre l'ingresso 4 è al livello logico LOW (valore 0), l'ingresso analogico ha valore 0015 (su una scala che va da 0 a 1023) e che l'ultimo salvataggio dei dati è avvenuto alle ore 12 del 1 settembre.

SCRITTURA DEL FILE CON IL VALORE DEGLI INGRESSI per la stazione n.333 1234567 con password 123456													
URL da usare: http://sicur.altervista.org/scrivi.php?numtel=333123457&ingr1=1&ingr2=1&ingr3=0&ingr4=0&analog=0015&pwd=123456													
Carattere Iniziale	Valore INGR 1	Separatore	Valore INGR 2	Separatore	Valore INGR 3	Separatore	Valore INGR 4	Separatore	Valore Analog. (0 ÷ 1023)	Separatore	Data formato GG/MM/AAAA	Spazio Separatore	Ora formato HH:MM:SS
$	1	%	1	%	1	%	0	%	0015	%	01/09/2014		12:00:00

LETTURA DEL FILE DEGLI INGRESSI per stazione n.333 1234567 con password 123456													
URL da usare: http://sicur.altervista.org/leggi.php?numtel=3331234567&pwd=123456													
Carattere Iniziale	Valore INGR 1	Separatore	Valore INGR 2	Separatore	Valore INGR 3	Separatore	Valore INGR 4	Separatore	Valore Analog. (0 ÷ 1023)	Separatore	Data formato GG/MM/AAAA	Spazio Separatore	Ora formato HH:MM:SS
$	1	%	1	%	1	%	0	%	0015	%	01/09/2014		12:00:00

Figura 4: Contenuto del file 3331234567_IN.txt e le URL per la lettura e scrittura

Il valore analogico è compreso nel tipico intervallo di un campionatore a 10 bit i cui valori analogici sono compresi tra 0 e 5V con passo 4,9mV, pertanto se il nostro segnale analogico ha escursioni maggiori dobbiamo elaborare un circuito di condizionamento del segnale che lo faccia rientrare nel range di funzionamento di Arduino. Un esempio banale può essere dato da un valore con escursione del segnale da 0 a 12V, pertanto risolvendo con un semplice partitore di tensione abbiamo che il limite dei 12V deve corrispondere a 5V pertanto devono essere rispettate la seguenti relazioni:

$$12V \cdot \frac{R_2}{R_1 + R_2} = 5V \quad \text{e} \quad 12V \cdot \frac{R_1}{R_1 + R_2} = 12V - 5V \quad \text{da cui} \quad R_1 = 14k\Omega \ \text{e} \ R_2 = 10k\Omega$$

Al fine di garantire un corretto funzionamento è indispensabile che le masse dei segnali 12V e 5V di Arduino siano allo stesso potenziale, ovvero cortocircuitati.
Tornando alla scrittura del file sul server è utile precisare che questa funzione (scrivi.php) verrà richiamata esclusivamente da Arduino per salvare lo stato attuale degli ingressi e verrà richiamata sia nel caso in cui varia lo stato di un ingresso e sia allo scadere di un timer impostato in maniera tale da prevenire eventuali attacchi da parte di dispositivi jammer (se il file non viene aggiornato continuamente si può supporre che ci sia un attacco al sistema). La lettura dello stato degli ingressi avviene, invece, ad opera del software di gestione richiamando l'URL dello script leggi.php del server e passando il numero di telefono e la password. Lo script opera aprendo il file in modalità "r" (modalità solo lettura partendo dall'inizio del file) e costruisce una stringa con i caratteri che legge all'interno del file "numtel_IN.txt": tale stringa verrà restituita tramite un comando echo a fine script.

Librerie o comandi AT.

Esistono in rete molteplici librerie per semplificare la gestione del modulo SIM900 ed il risparmio di energie è notevole: anziché comunicare attraverso comandi in standard AT si usano delle funzioni più ad alto livello che risultano molto più comprensibili. Tra le varie librerie merita di essere menzionata la libreria Open Source di GSMLIB.ORG sia perché è Open Source e sia perché è modulare, nel senso che si possono importare solo i moduli di

cui necessitiamo al fine di risparmiare risorse in termini di RAM e sia per la semplicità e potenzialità: la libreria si può scaricare dal sito https://code.google.com/p/gsm-shield-arduino/ o dal sito ufficiale http://www.gsmlib.org/ . La scelta adottata da me è stata quella di utilizzare i comandi AT per avere un maggiore controllo: è possibile trovare in rete molta documentazione sui comandi AT, ma consiglio di leggersi la guida ai comandi AT della SimCom (la si trova all'indirizzo http://wm.sim.com/downloaden.aspx?id=2986).

Il firmware di Arduino

Lo sketch contenuto nel file sicurduino.ino si apre inizializzando i PIN RX e TX che comunicheranno via seriale emulata con lo shield creando un'istanza della classe SoftwareSerial e configurando i PIN che saranno gli INPUT digitali e analogici. Vengono poi create le variabili che memorizzano lo stato precedente degli input che serviranno per stabilire se un ingresso ha variato il suo stato o meno. Viene infine creata un'ultima variabile globale: url che è una stringa che contiene l'url da richiamare nella funzione comunica(). Nella sezione setup dello sketch vengono impostati i PIN digitali in modalità INPUT_PULLUP in cui vengono abilitati dei resistori interni di pull-up che mantengono in HIGH il valore delle porte fino a che non interviene un cortocircuito verso massa determinato, magari, da un sensore che in caso di allarme richiuda il contatto verso GND.
Lo sketch prosegue eseguendo in loop due routine: verificaIngressi() e verificaTimer(). La funzione verificaIngressi() confronta il valore attuale degli ingressi con il valore salvato nelle variabili di stato: se il valore salvato differisce da quello attuale si procede a modificare la stringa dell'url (la variabile url) da richiamare al momento della trasmissione dati alla centrale. Si costruiscono due stringhe: una stringa viene ottenuta tramite la funzione substring e contiene il vecchio valore dell'ingresso; l'altra stringa viene costruita con il nuovo valore dell'ingresso. Si procede mediante la funzione replace della classe String ad attuare la sostituzione delle due stringhe all'interno della variabile url.

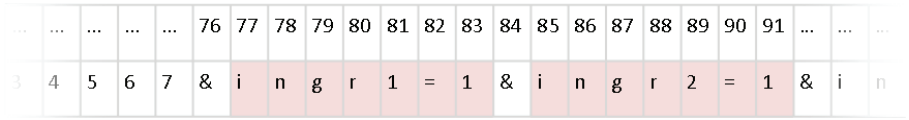


Figura 5: Caratteri estratti dalla funzione substring

Per ciò che riguarda il valore analogico viene trasformato in una stringa di dimensione fissa di 4 caratteri i cui valori nulli sono sostituiti da "0" (ad esempio un valore "15" viene trasformato in "0015"). Infine viene richiamata la routine comunica() che "naviga" l'url in cui sono contenuti i dati da passare al server tramite metodo GET. La routine comunica() è composta unicamente da stringhe di comandi che vengono passati al SIM900: si tratta di una sequenza di comandi AT che configurano ed abilitano il servizio GPRS e che effettuano una richiesta http all'url desiderato.
La seconda routine richiamata nel loop è verificaTimer() che sfrutta la funzione millis() la quale restituisce il numero di millisecondi trascorsi dall'accensione di Arduino: viene verificata la differenza tra i millisecondi trascorsi al momento della chiamata della funzione millis() e l'ultimo valore salvato nella variabile prevMillis; se la differenza tra i due valori supera il valore della variabile intervallo allora significa che è necessario sincronizzarsi con la centrale al fine di evitare un messaggio di timeout nel software di gestione installato sul PC. Il timer millis() va in overflow ogni 49,7 giorni circa ricominciando il conteggio da 0 e tale situazione è trasparente al codice.

Il software di gestione

Per semplificare e velocizzare lo sviluppo del software è stato utilizzato come linguaggio di programmazione il Visual Basic .NET in ambiente di sviluppo Microsoft Visual Studio 2008 Express Edition (versione gratuita): tale scelta è, ovviamente, personale e non vincolante. Senza entrare nello specifico, dato che il codice è ampiamente commentato, si illustreranno le principali funzioni e routine del software.

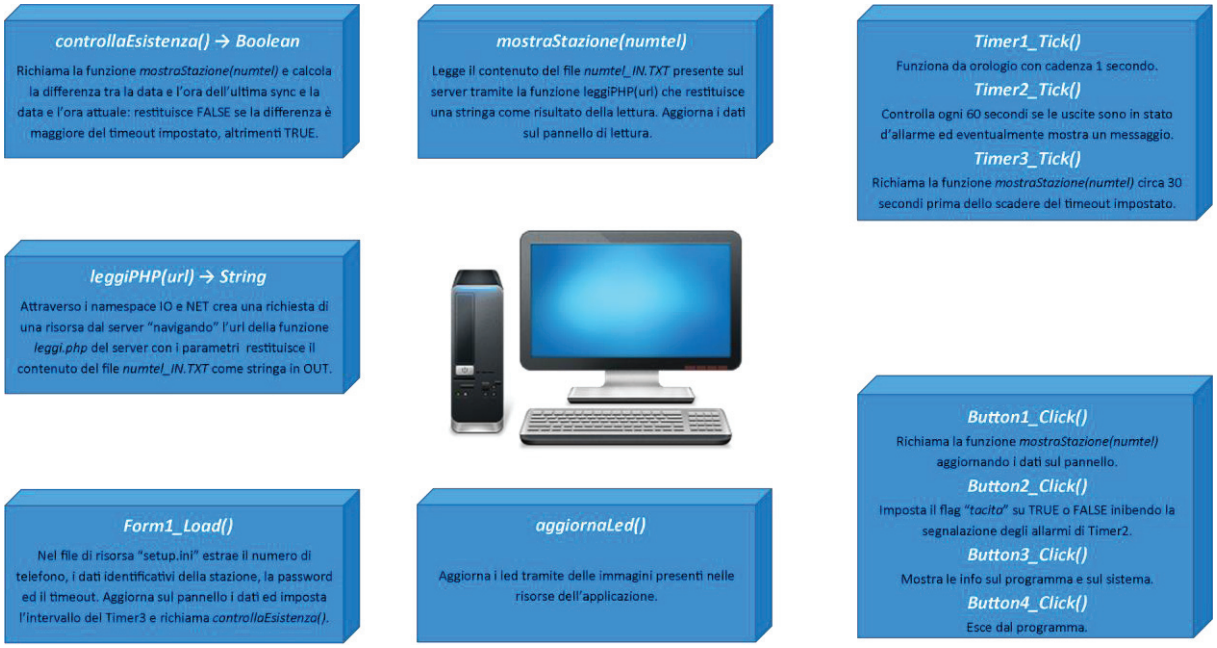


Figura 6: Procedure e Funzioni del Software realizzato in VB.NET 2008

All'avvio del programma verrà eseguito il codice all'interno della Sub Form1_Load e si inizia a leggere all'interno del file setup.ini che è stato inserito come risorsa del programma. Il file setup.ini contiene il numero di telefono, il titolo, l'indirizzo e la password della stazione da monitorare, tutte separate dal carattere "^". Viene richiamata la Sub mostraStazione che scarica dal server i dati degli ingressi e dell'ultima sincronizzazione tramite la chiamata dello script leggi.php presente sul server. Si procede ad impostare il Timer3 affinché controlli 30 secondi prima del tempo di timeout se la stazione sta correttamente comunicando con la centrale; a tal fine viene chiamata la Sub controllaEsistenza che, in caso di mancata comunicazione entro il termine massimo di timeout, avverte l'utente aprendo una MessageBox di avviso. La routine mostraStazione è realizzata tramite una chiamata alla Sub leggiPHP che crea una richiesta di pagina web attraverso le funzioni del namespace System.Net e che restituisce una stringa contenente la risposta dal server. La risposta è, quindi, la stringa contenuta nel file numtel_IN.txt che sarà del tipo

$1%1%1%0%0015%01/09/2014 12:00:00

con il significato visto in precedenza.
La Sub controllaEsistenza si realizza recuperando dal server la data e l'ora dell'ultima sincronizzazione e poi la confronta con la data e l'ora attuale: se la differenza è maggiore del tempo di timeout restituisce il valore False il quale indica che la stazione non sta trasmettendo alla centrale.

La diretta conseguenza della scelta di non usare un database engine porta ad utilizzare le funzioni della classe String (come ad esempio Substring e Concat) al posto di richiamare i campi di una tabella che risulterebbe sicuramente più leggibile a livello di codice ma appesantirebbe il progetto nel suo complesso (sarebbe necessario installare il pesante MySqlServer o altro database engine).

Figura 7: Una schermata del software

Conclusioni

Volutamente il sistema periferica-centrale è stato semplificato ed alcuni aspetti non sono stati approfonditi mentre altri sono stati completamente trascurati. Espandere il sistema è possibile, anzi è consigliabile, soprattutto se l'utilizzo non è del tipo didattico ma semi-professionale. Ad esempio si potrebbe rendere personalizzabile la stazione da monitorare sul software del PC, cambiare la password sul sistema Arduino-SIM900 tramite SMS di configurazione (memorizzando i dati nella EEPROM interna di Arduino ad esempio), controllare la risposta dei comandi AT del SIM900 per verificare se si può procedere con il comando successivo, aumentare la sicurezza del server ed altre migliorie.

Pin	Descrizione
GND	Massa di alimentazione
+5V	Positivo di alimentazione (min. 2A)
D0	RX in comunicazione UART Hardware
D1	TX in comunicazione UART Hardware
D7	RX in comunicazione UART Software
D8	TX in comunicazione UART Software
D9	Comando di Power Software

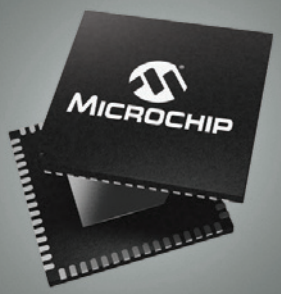
Tabella1: Interfaccia dello SHIELD EFCOM PRO GPRS/GSM

L'Intelligent Design

parte dagli Intelligent Analog PIC® Microcontroller

INTELLIGENT
ANALOG

La progettazione analogica è difficoltosa, e assorbe prezioso tempo di sviluppo. Gli intelligent PIC® MCU di Microchip integrano funzionalità analogiche come un Analog-to-Digital Converter di elevate prestazioni, Digital-to-Analog Converter, e op amp, offrendo interfacce di facile utilizzo che semplificano la progettazione analogica. Una soluzione single-chip che permette di ridurre il rumore di sistema, offre un maggiore throughput, e al contempo riduce drasticamente i tempi e costi di progettazione.



Applicazioni

- Sensori ambientali di qualità
- Apparecchiature medicali portatili
- Apparecchiature Industriali
- Conversione di potenza
- Efficienza motori
- Illuminazione
- Misurazione e controllo della potenza
- Apparecchiature di energy harvesting
- Controllo di inverter solari

microchip DIRECT
www.microchipdirect.com

MICROCHIP

microchip.com/intelligentanalog

Introduzione ai sistemi

ASIC, FPGA e CPLD

Di Massimiliano Miocchi

La migrazione di un progetto basato su FPGA o CPLD in un progetto basato su ASIC può sembrare a prima impegnativo ed oneroso per un team di progettazione, ma un'attenta pianificazione possono facilitare notevolmente questo processo

Per trasformare un progetto digitale realizzato tramite FPGA in un equivalente progetto adatto alla produzione di massa vi sono varie soluzioni: Soluzioni a basso costo come i circuiti ASIC (Application-Specific Integrated Circuit) strutturati, i circuiti integrati basati su celle standard e gate array di gate offrono prestazioni più elevate, un minore consumo energetico, una migliore integrazione e una maggiore immunità ai disturbi elettromagnetici. La migrazione di un progetto basato su FPGA in un progetto basato su ASIC può sembrare a prima vista impegnativo per un team di progettazione, ma un'attenta pianificazione possono facilitare notevolmente questo processo.
Progettare un nuovo prodotto sfruttando la tecnologia FPGA permette di introdurre rapidamente eventuali modifiche di progetto direttamente a livello hardware. Quando il codice del progetto è stabile e il dispositivo è pronto per la produzione, la migrazione da un FPGA a un ASIC può comportare una riduzione fino al 50% il costo unitario di produzione.
I sistemi più complessi potrebbero richiedere diversi circuiti FPGA per realizzare un prototipo di un intero progetto SoC(System-on-a-Chip). Gli ASIC offrono una maggiore densità di gate e prestazioni di base maggiori rispetto agli FPGA.
Ciò consente di combinare più FPGA in un unico SoC ASIC, risparmiando spazio prezioso sulla scheda di conseguenza abbattendo i costi. Se l'architettura non permette una piena integrazione, è possibile realizzare una sostituzione funzionale diretta e veloce, producendo un ASIC multifunzionale in grado di emulare i diversi circuiti FPGA. Questa soluzione ottimizza i risparmi riducendo i costi per la trasformazione in tecnologia ASIC. Ci sono diversi aspetti da considerare nel flusso produttivo di questa tecnologia, un aspetto importante è sicuramente la produzione cn flusso parallelo:

Flusso produttivo parallelo

I flussi di progettazione parallela permettono di abbreviare i tempi di sviluppo dell'ASIC. La prototipazione FPGA permette
ai team tecnici di verificare la funzionalità prima della realizzazione fisica del circuito ASIC e dà inoltre la possibilità di collaudare in maniera tempestiva l'hardware e il software utilizzati. In questo flusso di progettazione, il codice RTL (Register Transfer Level) viene sviluppato e adattato all'FPGA e contestualmente dato al fornitore di ASIC per essere analizzato e rivisto. Ciò permette al fornitore di ASIC di fornire raccomandazioni ed effettuare modifiche al codice al fine di aumentarne la robustezza. Gli script di temporizzazione e le architetture di clock vengono sviluppate sia per l'FPGA che per l'ASIC. Il package dell'ASIC viene scelto valutando anche le problematiche di integrità del segnale e della dissipazione di potenza. Il circuito stampato può essere progettato in questa fase tenendo conto sia della presenza degli FPGA, che assorbono più potenza, sia

della successiva migrazione più circuiti ASIC fisicamente più piccoli e più efficienti sotto l'aspetto energetico.
Abbiamo fin qui accennato ad una delle più avanzate tecnologie in uso, se pur complesse le ASIC. Non necessariamente si deve ricorrere a tale tecnologia , almeno non in questo contesto dove analizzeremo in modo semplice e chiaro la struttura di una FPGA e le differenze funzionali con le CPLD. Il lettore tenga bene in mente che oggi quando si parla di tecnologia FPGA implicitamente ci si proietta in un sistema di progettazione completamente diverso, non più statico come può essere l'uso degli IC ma dinamico come i DSP o ***digital signal processor***.
Tutti noi abbiamo trovato difficoltà non sempre superabili nel progettare sistemi sequenziali o combinatori dove l'uso di eccezioni o processi di confronto assumono complessità notevoli; bene, in queste circostanze è consigliabile l'uso di FPGA o microcontrollori rendendo la vita facile al progettista per cosi dire...
In definitiva gli FPGA permettono di commercializzare rapidamente un prodotto e costituiscono un efficiente strumento di prototipazione, ma l'elevato costo degli FPGA di fascia medio-alta può essere proibitivo per passare a una produzione di massa. Con una corretta pianificazione durante la fase di sviluppo, la migrazione da FPGA ad ASIC può essere perfezionata rapidamente senza ulteriori sforzi. Vediamo ora come si è arrivati alle attuali tecnologie DSP,CPLD e FPGA.
Prima dell'avvento sui mercati mondiali delle tecnologie a microprocessore la tecnologia del tempo si presentava molto complessa in virtù del massivo utilizzo delle logiche combinatorie e sequenziali demandate ai single chip. Questo oltre a portare un dispendio di risorse portava con se anche un grande spreco nella progettazione in se, sia in termini di costi che di componentistica.

L'elettronica digitale PRIMA dei microcontrollori

Le tecnologie prima degli anni 90 erano per l'appunto le famiglie logiche ,con le quali si realizzava tutto, dai sistemi di controllo a quelli di trasmissione e recezione e non senza complicazioni.

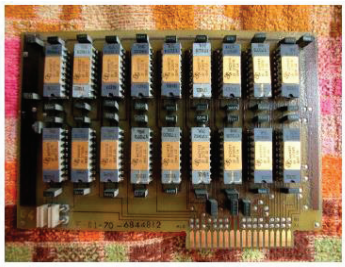
Prima dei Microcontrollori le macchine di calcolo erano come in figura:

Anche le centraline auto che oggi sono super automatiche con protocolli di trasmissione

come il CanBus, all'epoca erano cosi:

Centralina auto: ECU

per non parlare delle schede RAM, circuiterie molto complesse:



Tutto questo ha fatto da generatore per avviare un processo di evoluzione tecnologica dettato dalle pressanti e difficili esigenze di mercato.

L'ERA DELLE LOGICHE TTL

La tecnologia delle logiche discrete TTL ottenne il battesimo ufficiale quando, agli inizi degli anni '60, la Texas Instruments introdusse la già accennata famiglia 74 di circuiti integrati di tipo standard logic per supportare i programmi di esplorazione spaziale, in particolare lunare, della NASA.

La famiglia 74 (di cui in figura 1.2 è rappresentato un esemplare) comprendeva le porte logiche elementari, come ad esempio il quadruplo NAND 7400 (figura 1.3), oltre alle varie tipologie di flip-flop (tra cui il celebre 7474), i contatori (come il 7493), i sommatori binari e altre semplici funzioni. I primi integrati erano fabbricati nei package dual-in-line a 14 e 16 pin.
I successivi progressi nel packaging e l'introduzione di dispositivi a livelli via via crescenti di integrazione consentirono ai progettisti di minimizzare drasticamente gli ingombri sui circuiti stampati e di ridurre il numero di componenti complessivo (il cosiddetto chip count), fornendo nel contempo un comodo sistema "ad incastro" del tipo a blocchetti Lego.
Con il trascorrere degli anni nuove famiglie venivano introdotte continuamente, atte a soddisfare le necessità di aumento delle prestazioni in frequenza, minori consumi e riduzione dei costi. Il risultato fu la proliferazione di oltre trenta differenti famiglie 74, ciascuna con specifiche caratteristiche. Attraverso gli anni '70 le varianti delle famiglie TTL si diffusero rapidamente e, di pari passo, neaumentò la complessità. Nei primi anni '80 era presente sul mercato un' estrema varietà di tipologie : TTL, S, LS, AS, F, ALS, CD4000,

HC, HCT, BCT, AC, ACT, FCT, ABT, LVT, AHC, AHCT, ecc.., tali da costringere i progettisti a destreggiarsi a fatica tra le caratteristiche di ognuna e le reciproche compatibilità elettriche per poter al meglio realizzare le proprie applicazioni.

NASCITA E AVVENTO DELLE CPLD

Mentre la gamma dei dispositivi 74 si espandeva, cominciava ad apparire una nuova tendenza rivoluzionaria nella forma dei componenti logici programmabili, detti PLD (Programmable Logic Devices). Oltre a consentire all' utilizzatore la caratteristica della programmabilità, la prima generazione di questi chip poteva rimpiazzare fino a una diecina di integrati logici discreti. In seguito, con l'avanzare del processo di integrazione, le PLD furono in grado di sostituire sempre più numerose funzioni logiche standard. Oggi, le più avanzate CPLD (Complex PLD) contengono decine di migliaia di gates e la loro capacità equivalente corrisponde a quella di centinaia di dispositivi della serie 74.

Ecco come ci si spinse oltre la logica fissa.

Esigenze tecniche e di mercato

Le richieste di flessibilità e integrazione hanno generato la necessità di dover raggruppare:

- Parallelismo delle operazioni
- Numero delle variabili di controllo
- Velocità di risposta
- Modularità del sistema
- Affidabilità del controllo
- Flessibilità dei componenti
- Reti logiche
- Logiche programmabili
- Controlli digitali e analogici

Se ne evince:
Riduzione dimensioni, riduzione costi, flessibilità, programmabilità
A questo punto è lecito osservare come il microcontrollore cominci a divenire indispensabile la dove le variabili in gioco erano molte, quindi complesse da controllare. Da qui si capisce come un microcontrollore sia efficace nello svolgere determinate funzioni come:

- Comunica con l'esterno solo mediante porte di input/output
- Esegue esclusivamente operazioni logiche, aritmetiche e di controllo
- Elabora sia dati prodotti internamente che provenienti da dispositivi esterni
- Totalmente dipendente da elementi periferici
- Non presenta elementi per la memorizzazione permanente dei dati

Il microcontrollore racchiude tutte le caratteristiche del microprocessore aggiungendo le

possibilità di:

- Comunicazione diretta con dispositivi esterni integrando periferiche interne
- Memorizzazione di dati o programmi
- Effettuare operazioni di controllo, ricezione ed elaborazione segnali
- In generale non necessitano ulteriori aggiunte di memoria RAM oltre a quella integrata
- Eseguono esclusivamente le operazioni legate al firmware con il quale sono stati programmati

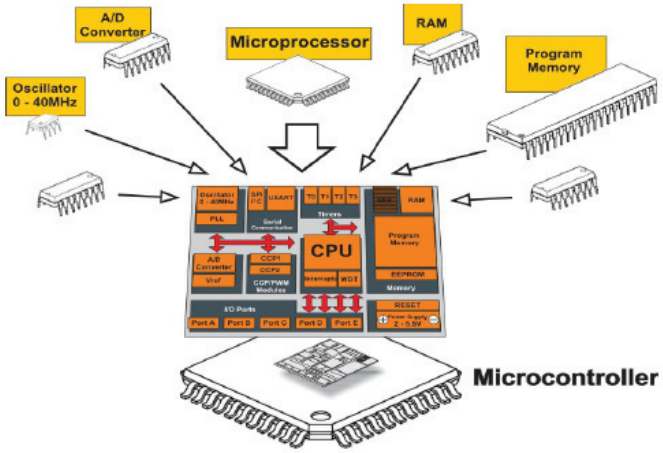
Microcontrollore e Microprocessore

Microcontrollore:

- Microprocessore
- Linee I/O
- Memoria Flash
- Convertirore A/D
- Timer
- USART
- SPI
- PWM
- ...

Microprocessore:

- Unità di calcolo
- Unità di controllo
- Memoria istruzioni e calcolo



Il microcontrollore è un'estensione del microprocessore, le sue caratteristiche lo avvicinano molto ad un computer completo. In definitiva il microcontrollore è un “sistema” a microprocessore realizzato con una logica di ottimizzazione del rapporto prezzo/prestazioni. A differenza del microprocessore ha un esteso campo di impiego che può spaziare dai più semplici oggetti di utilizzo quotidiano a complesse architetture in ambito medicale e industriale. Ha una capacità di calcolo relativamente limitata ed esegue esclusivamente una serie di operazioni contenuta all'interno del programma o firmware caricato nella memoria.

Le prime logiche programmabili nate negli anni ‘80 offrivano la possibilità di manipolare migliaia di gate di logica programmabile nelle tecnologie allora presenti. Questi dispositivi si indirizzavano ai progettisti che ambivano ad integrazioni logiche e volevano evitare i rischi connessi alle rigidità dei gate array ,ove potevano farne a meno. Inoltre,il tempo di ingresso nel mercato, dalla fase di stesura delle specifiche, era divenuto un requisito critico per la competitività sul mercato stesso. Sfruttando la riduzione del ciclo progettuale

che ne scaturiva con questo metodo di lavoro, i vari progettisti riuscivano ad introdurre nuovi prodotti sempre più rapidamente. Inizialmente questi dispositivi venivano chiamati High Density PLD ed erano considerate un'estensione logica delle logiche programmabili allora presenti (PAL e GAL). Essi venivano utilizzati in modo simile, cioè integrazione di logica sparsa (glue logic) precedentemente implementata usando circuiti integrati TTL e sostituzione di gate array di bassa densità.Pian piano con l'avanzare del tempo e con l'introduzione di nuove famiglie di componenti, nel tentativo di distinguere le differenti architetture di PLD, le varie case produttrici proposero il termine Field Programmable Gate Array (FPGA). Attualmente il processo tecnologico a 28nm ha permesso alle grosse case produttrici come ad esempio la Xilinx, di produrre FPGA ad elevatissimo numero di celle logiche programmabili: ad esempio l'ultimo arrivato in casa Xilinx è il Virtex 7 il quale presenta ben 2 milioni di celle logiche, 8 porte PCIe GEN3 e la possibilità di disporre di 1200 pin di I/O..e molto altro!

Struttura interna di una FPGA

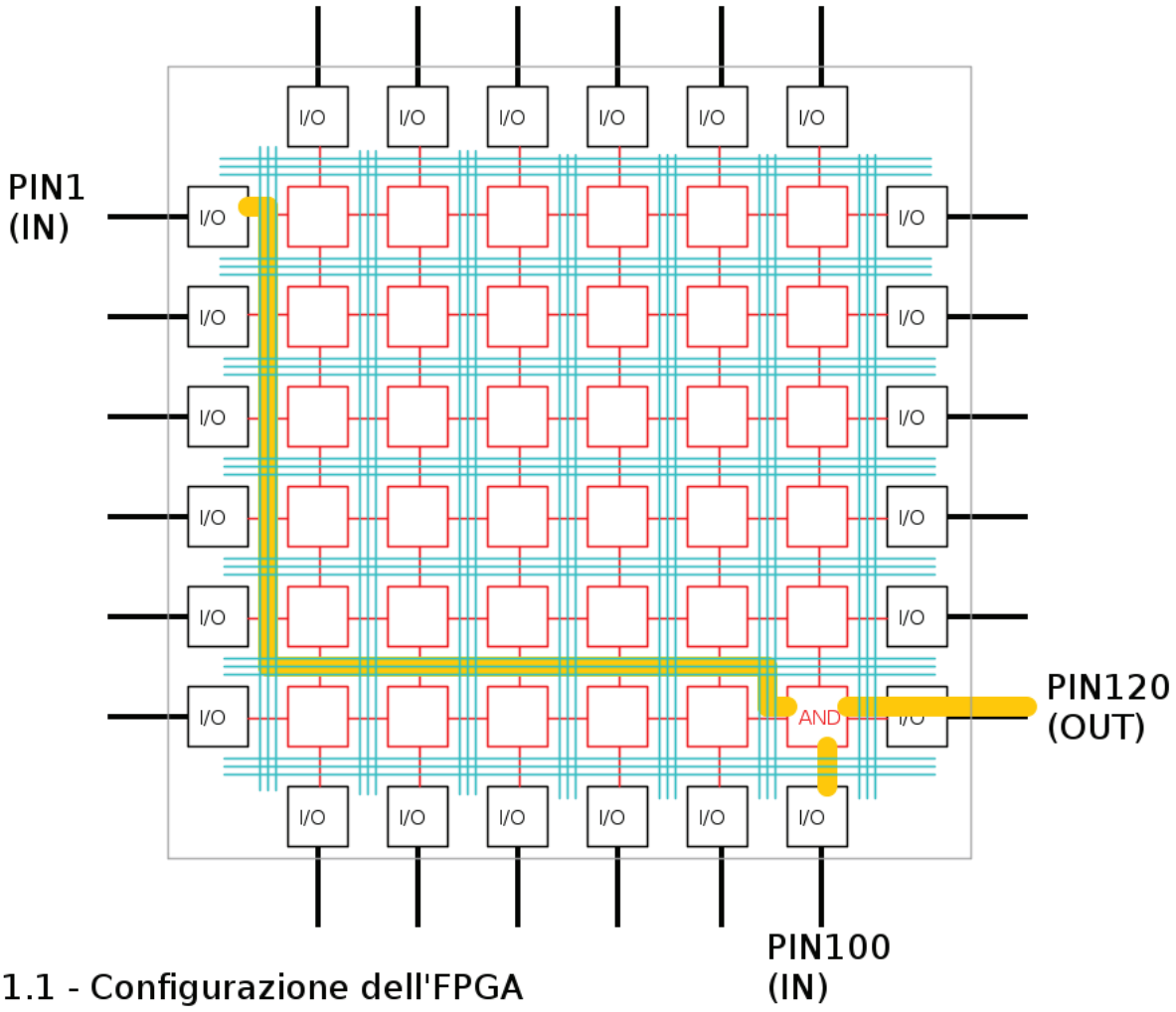


Fig. 1.1 - Configurazione dell'FPGA

Ogni FPGA viene realizzata attraverso una matrice di blocchi logichi configurabili o **Configurable Logic Blocks** (CLB), i quali vengono connessi attraverso delle opportune

Interconnessioni (piste metalliche) sia tra di loro che con pin di I/O per comunicare con segnali esterni all'FPGA. Analizziamo prima i CLB: ogni elemento logico della matrice viene interconnesso a quelli circostanti tramite due canali: uno verticale e uno orizzontale. Nei punti in cui il canale orizzontale incrocia quello verticale, vengono posizionati degli elementi contenenti la logica programmabile necessaria a gestire le connessioni: gli switch modules. Quindi grazie a tali elementi, è possibile realizzare un certo numero (finito) di funzioni logiche e il vantaggio sta tutto nel poter riprogrammare opportunamente le connessioni tra le varie CLB in modo tale da cambiare la funzione logica. Il limite ovviamente è dovuto al fatto che i CLB e i pin di I/O sono in numero finito quindi la funzione o le funzioni avranno un numero di operazioni finito (ad esempio al più non possiamo realizzare più di 10 funzioni che fanno quattro somme tipo Y=A+B+C+D ecc..).

Per non appesantire troppo l'articolo descriviamo brevemente la struttura di un CLB: Ogni CLB è costituito a sua volta da un altro elemento logico di base chiamato BLE. Il BLE è composto a sua volta da elementi logici combinatori e sequenziali, in pratica una Look Up Table(a 4 ingressi nell'esempio in figura) e un flip flop D. Il motivo per il quale si utilizza questa tecnica di raggruppare le LUT in BLE e poi in CLB è dovuto essenzialmente al fatto che facendo in questo modo si riducono i tempi di ritardo medi dovuti alla propagazione del segnale nelle varie interconnessioni. Le interconnessioni: Come già detto in precedenza i vari elementi logici sono connessi tra loro tramite due canali: ciascuno di questi due canali viene realizzato tramite delle piste metalliche le quali vengono anche dette "wire segment": ogni interconnessione si misura a seconda di quante CLB essa attraversa.

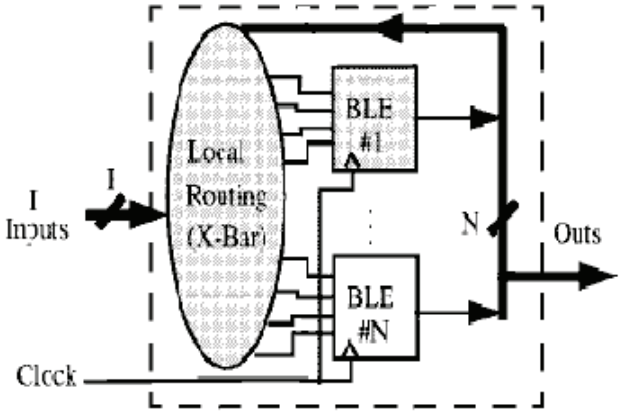


Figura 1.3a Blocco logico (CLB)

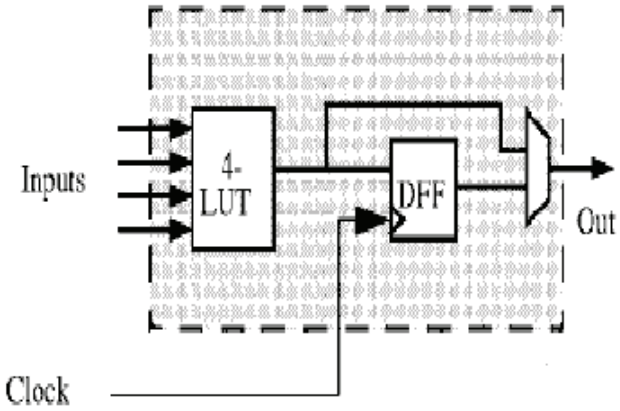


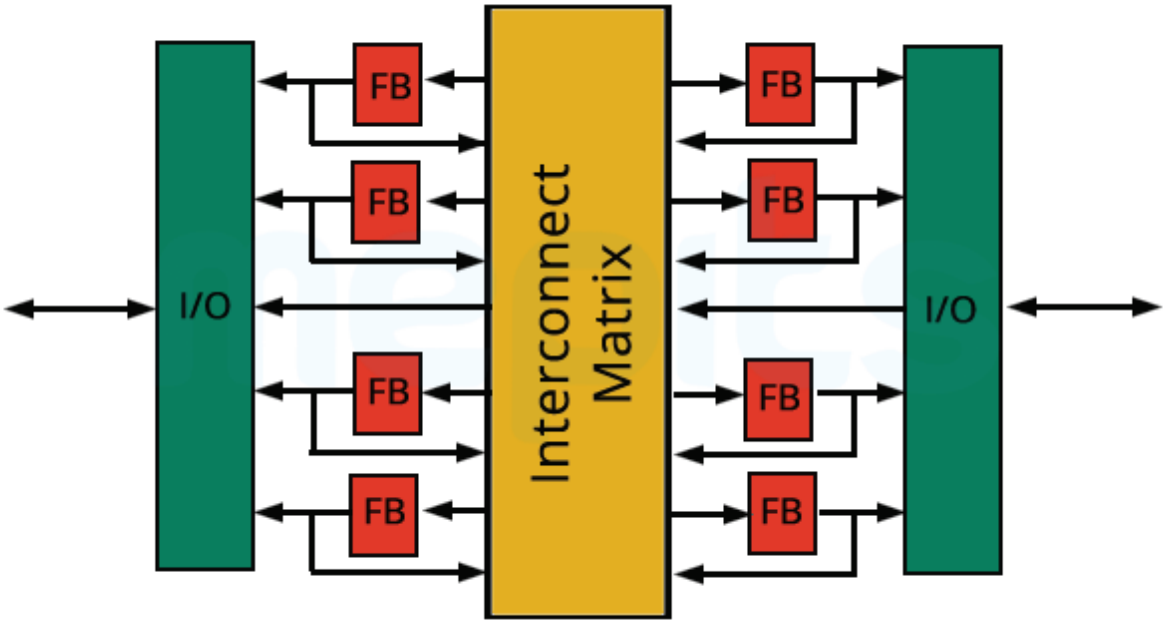
Figura 1.3b *Elemento logico di base (BLE)*

Il lettore tenga presente che qualsiasi sistema digitale può realizzarsi utilizzando i quattro elementi fondamentali: le porte NOT,AND e OR e i flip-flop. Disponendo di un circuito integrato in cui sono realizzati già dal costruttore tali elementi fondamentali, avendo invece la libertà di connetterli in base alle particolari specifiche volute, consente la realizzazione un qualunque sistema digitale. anche complesso, e fabbricato utilizzando un unico circuito integrato il cui test di funzionalità è stato già eseguito. Nasce proprio così l'idea di

dispositivi programmabili dall'utente che, grazie all'enorme progresso in termini sia di costo che di livello di integrazione, ha portato a interessanti progressi dei PLD (*Programmable Logic Devices*). Spesso, più propriamente, tali dispositivi vengono denominati come programmabili sul campo FPD, (Field Programmable Device) nel senso che è l'utente finale a poter programmare la funzione specifica; I PLD nascono per applicazioni di elettronica digitale anche se recentemente molti sono i prodotti che cercano di portare la stessa filosofia di progettazione e architettura in ambiente analogico.

Oltre alla tecnologia FPGA abbiamo le CPLD o EPLD, nascono prima sempre sull'ona della medesima filosofia di progetto.

Sono dispositivi che presentano una complessità maggiore sono generalmente riprogrammabili . Questi consistono tipicamente in più Gate Array Logic integrati su un solo chip. Nella struttura si possono distinguere i blocchi in cui può essere configurata una funzione logica (CFB, Configurable Function Block o GLB, Generic Logic Block) e una matrice di interconnessione programmabile centralizzata (GRP, global routing Poo). La similitudine con le FPGA è forte, spesso e volentieri conviene utilizzare le CPLD in applicazioni di reti complesse ove non sarebbe giustificata la FPGA. La struttura di una CPLD è rappresentata in figura:



CONVENIENZA DELLE LOGICHE PROGRAMMABILI

I componenti logici discreti, a lungo considerati i cavalli di battaglia dell'industria dei semiconduttori, sono stati per parecchi anni avvantaggiati dai bassi costi, rispetto alle prime logiche programmabili comparse agli inizi sul mercato.
Tuttavia, oggi le cose sono cambiate radicalmente; i progressi compiuti negli anni più recenti nel campo delle tecnologie di costruzione delle CPLD hanno abbassato i costi di produzione di questi dispositivi a un livello tale da farli divenire alternative molto appetibili

rispetto alle logiche fisse discrete.
Ma, al di là delle semplici ragioni di costo, vi sono altri fattori particolarmente rilevanti che rappresentano notevoli vantaggi a favore delle CPLD : la flessibilità costituita dalla riprogrammabilità, la rapidità nell' apportare le modifiche, la riduzione dell'area occupata sullo stampato, la superiore affidabilità rispetto alle logiche discrete, il ridotto time-to-market, solo per citare i più importanti.

Osservazioni

Il lettore tenga presente che lo studio e l'applicazione di tecnologie FPGA o CPLD è strettamente legata alla conoscenza dei microcontrollori, inoltre sono chip molto delicati facili da danneggiare se non si osservano regole ben precise nell'utilizzarle. Per il resto sono affascinanti consentono di realizzare combinazioni di sistemi complessi con estrema efficacia e velocità, si velocità parliamo di strutture capaci di viaggiare anche a centinaia di Mhz. Nella prossima puntata realizzeremo una demo board con a bordo una CPLD della Xilinx necessaria al prosequio del mini corso. Per chi fosse curioso può documentarsi sul linguaggio vhdl, standard come il verilog per la programmazione hardware di questi gioiellini.

Interfacciamento Sensore di Pressione MPX4115AP con il PIC18F45K222

di Alberto Trasimeni

Esiste una ampia scelta di sensori di pressione assoluta con range di misurazione che va dai 15Kpa ai 115Kpa,tali sensori sono ideali per la realizzazione di barometri ed altimetri. In questo articolo prenderò in considerazione il sensore MPX4115AP della Freescale che risulta compensato in temperatura e calibrato.

Caratteristiche generali del sensore e del PIC

Il sensore MPX4115AP è un sensore di pressione assoluta di tipo analogico, quindi per leggere la misura di pressione fornita, è necessario fare uso di un convertitore analogico/digitale. Per interfacciarlo sfrutterò il PIC18F45K22,dotato di più convertitori analogico/digitale e tutti a 10 bit di conversione. Inoltre userò l'oscillatore interno del microcontrollore, alla frequenza di 8Mhz.Come sistema di sviluppo,per il progetto, mi servirò della EasyPIC7 della MikroElektronica e come ambiente di sviluppo software, adopererò il MikroC, nella sua ultima versione la 6.4.0.

Le caratteristiche principali del sensore sono:

- Massimo errore nella misura è del ±1,5% con temperatura compresa tra 0 e 85 gradi celsius.
- Ideale per applicazioni basate su microprocessore/microcontrollore.
- Applicazioni più ricorrenti realizzazione di altimetri e barometri

L'unità di misura della pressione fornita è Kpascal ma può essere facilmente convertita in millibar e Torricelli.

Il sensore si presenta fisicamente come in figura sottostante.

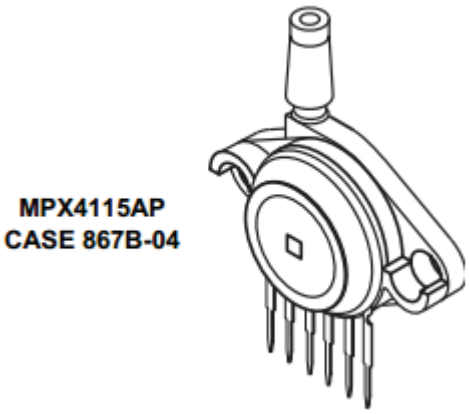


Figura 1: Case del sensore MPX4115

La struttura funzionale interna è costituita da un elemento sensibile alla pressione seguito

da un primo stadio, il cui scopo è quello di compensare la misura di pressione con la temperatura, mentre il secondo ha la funzione di bilanciare l'uscita verso massa, come mostra la figura sottostante.

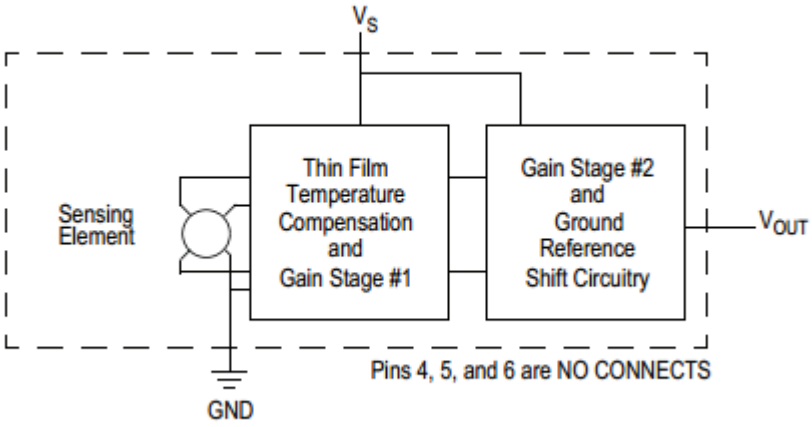


Figura 2: Struttura funzionale interna

La tabella riporta i valori limiti di pressione associati alla temperatura, per cui si ha il danneggiamento o una degradazione rispetto al normale funzionamento.

Parametrics	Symbol	Value	Unit
Overpressure(2) (P1 > P2)	P_{max}	400	kPa
Burst Pressure(2) (P1 > P2)	P_{burst}	1000	kPa
Storage Temperature	T_{stg}	-40° to +125°	°C
Operating Temperature	T_A	-40° to +125°	°C

1. T_C = 25°C unless otherwise noted.
2. Exposure beyond the specified limits may cause permanent damage or degradation to the device.

Tabella 1:Valori limiti

Characteristic		Symbol	Min	Typ	Max	Unit
Pressure Range(1)		P_{OP}	15	-	115	kPa
Supply Voltage(2)		V_S	4.85	5.1	5.35	Vdc
Supply Current		I_o	—	7.0	10	mAdc
Minimum Pressure Offset(3) @ V_S = 5.1 Volts	(0 to 85°C)	V_{off}	0.135	0.204	0.273	Vdc
Full Scale Output(4) @ V_S = 5.1 Volts	(0 to 85°C)	V_{FSO}	4.725	4.794	4.863	Vdc
Full Scale Span(5) @ V_S = 5.1 Volts	(0 to 85°C)	V_{FSS}	—	4.59	—	Vdc
Accuracy(6)	(0 to 85°C)	—	—	—	±1.5	$\%V_{FSS}$
Sensitivity		V/P	—	46	—	mV/kPa
Response Time(7)		t_R	—	1.0	—	ms
Output Source Current at Full Scale Output		I_o+	—	0.1	—	mAdc
Warm-Up Time(8)		—	—	20	—	mSec
Offset Stability(9)		—	—	±0.5	—	$\%V_{FSS}$

1. 1.0kPa (kiloPascal) equals 0.145 psi.
2. Device is ratiometric within this specified excitation range.
3. Offset (V_{off}) is defined as the output voltage at the minimum rated pressure.
4. Full Scale Output (V_{FSO}) is defined as the output voltage at the maximum or full rated pressure.
5. Full Scale Span (V_{FSS}) is defined as the algebraic difference between the output voltage at full rated pressure and the output voltage at the minimum rated pressure.
6. Accuracy (error budget) consists of the following:
   Linearity:Output deviation from a straight line relationship with pressure, using end point method, over the specified pressure range.
   Temperature Hysteresis:Output deviation at any temperature within the operating temperature range, after the temperature is cycled to and from the minimum or maximum operating temperature points, with zero differential pressure applied.
   Pressure Hysteresis:Output deviation at any pressure within the specified range, when this pressure is cycled to and from the minimum or maximum rated pressure at 25°C.
   TcSpan:Output deviation over the temperature range of 0° to 85°C, relative to 25°C.
   TcOffset:Output deviation with minimum pressure applied, over the temperature range of 0° to 85°C, relative to 25°C.
   Variation from Nominal:The variation from nominal values, for Offset or Full Scale Span, as a percent of V_{FSS} at 25°C.
7. Response Time is defined as the time for the incremental change in the output to go from 10% to 90% of its final value when subjected to a specified step change in pressure.
8. Warm-up is defined as the time required for the product to meet the specified output voltage after the Pressure has been stabilized.
9. Offset stability is the product's output deviation when subjected to 1000 hours of Pulsed Pressure, Temperature Cycling with Bias Test.

La tabella riporta le caratteristiche operative del sensore in condizioni di normale funzionamento.

Tabella 2: Caratteristiche operative del dispositivo

Il collegamento elettrico del sensore deve avvenire secondo lo schema sotto riportato. I due condensatori presenti sulla linea di alimentazione hanno lo scopo di filtrare il rumore presente su tale linea(di solito 100nf multistrato e 1.0us al tantalio).
È comunque necessario, dal momento che abbiamo a che fare con un sensore analogico, separare l'alimentazione e la massa di questo dal resto del circuito.Ciò può avvenire, nella fase realizzativa del pcb, predisponendo una alimentazione analogica ed una massa analogica, che per esempio chiameremo AVcc e AGnd.
Queste due linee dovranno essere splittate, con l'alimentazione generale e la massa generale, in prossimità dell'alimentatore del sistema, che rappresenta il punto ad impedenza più bassa presente nel circuito.

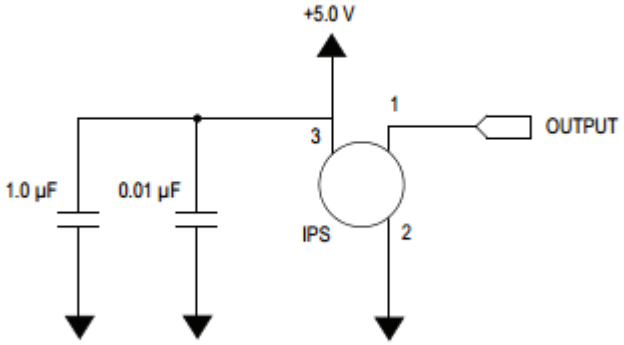


Figura 3: Collegamento elettrico del sensore

La relazione che esprime la tensione di uscita in funzione della pressione misurata, è data da:

$$V_{out}=Vs*(0.009*P-0.095)$$

in cui Vs è il valore della tensione analogica di alimentazione e P è la pressione rilevata in Kpascal.

Le relazioni che legano i Kpascal ai millibar e ai Torricelli sono:

mbar=Kpa*10

Torr=Kpa*7.5006

È quindi possibile ricavare la V_{out} in funzione di queste nuove unità di misura, ed avremo:

V_{out}=Vs*(0.009* mbar/10 -0.095)

V_{out}=Vs*(0.009*Torr/7.5006 -0.095)

Ipotizzando Vs=5.0v mediante il foglio elettronico Excel è possibile ricavare la caratteristica di trasferimento del sensore in Kilopascal, millibar e Torricelli, come mostrano le figure sottostanti.

	A	B	C	D	E	F
1	Kpascal	Risposta sensore MPX4115(V-Kpa)	millibar	Risposta Sensore MPX4115(V-mbar)	Torricelli	Risposta sensore MPX4115(v-Torr)
2	15	0,204	150	0,204	112,51	0,204
3	20	0,434	200	0,434	150,01	0,434
4	25	0,663	250	0,663	187,52	0,663
5	30	0,893	300	0,893	225,02	0,893
6	35	1,122	350	1,122	262,52	1,122
7	40	1,352	400	1,352	300,02	1,352
8	45	1,581	450	1,581	337,53	1,581
9	50	1,811	500	1,811	375,03	1,811
10	55	2,040	550	2,040	412,53	2,040
11	60	2,270	600	2,270	450,04	2,270
12	65	2,499	650	2,499	487,54	2,499
13	70	2,729	700	2,729	525,04	2,729
14	75	2,958	750	2,958	562,55	2,958
15	80	3,188	800	3,188	600,05	3,188
16	85	3,417	850	3,417	637,55	3,417
17	90	3,647	900	3,647	675,06	3,647
18	95	3,876	950	3,876	712,56	3,876
19	100	4,106	1000	4,106	750,06	4,106
20	105	4,335	1050	4,335	787,56	4,335
21	110	4,565	1100	4,565	825,07	4,565
22	115	4,794	1150	4,794	862,57	4,794

Figura 4: Caratteristica di trasferimento del sensore

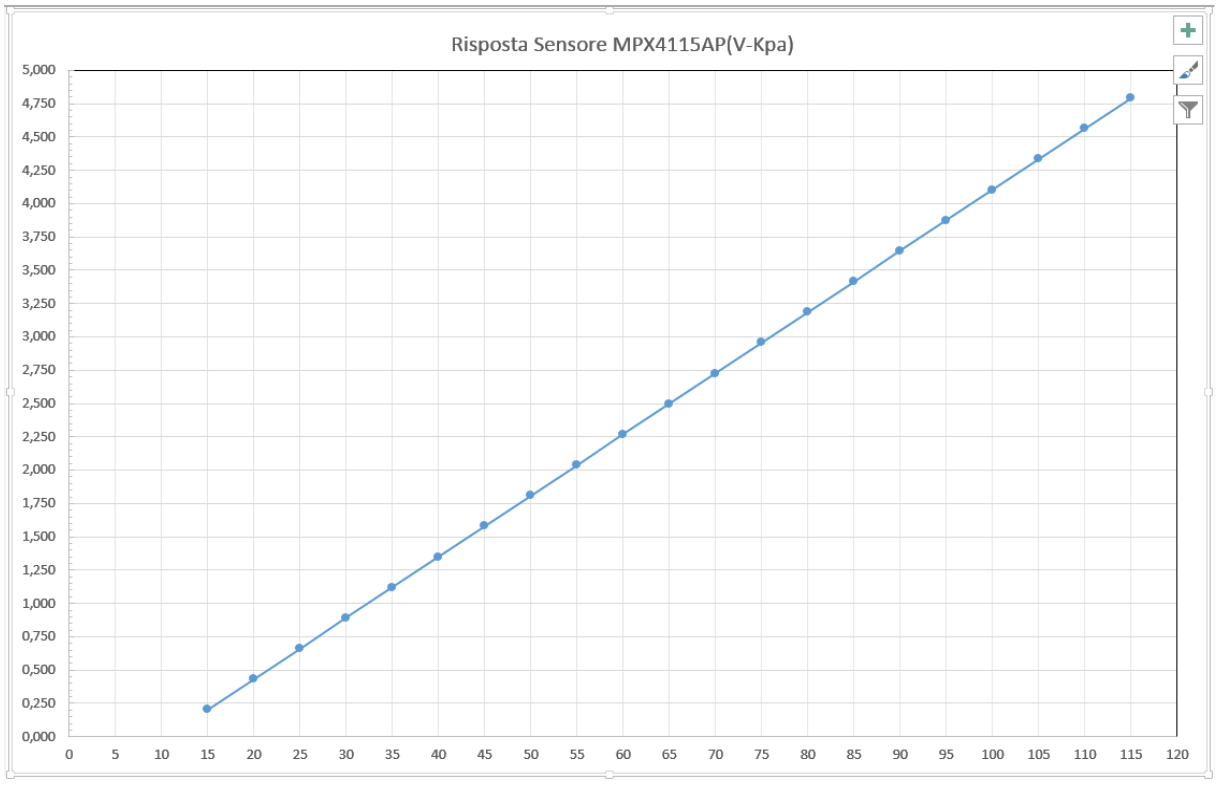


Figura 5: Risposta del sensore MPX4115(V-Kpa)

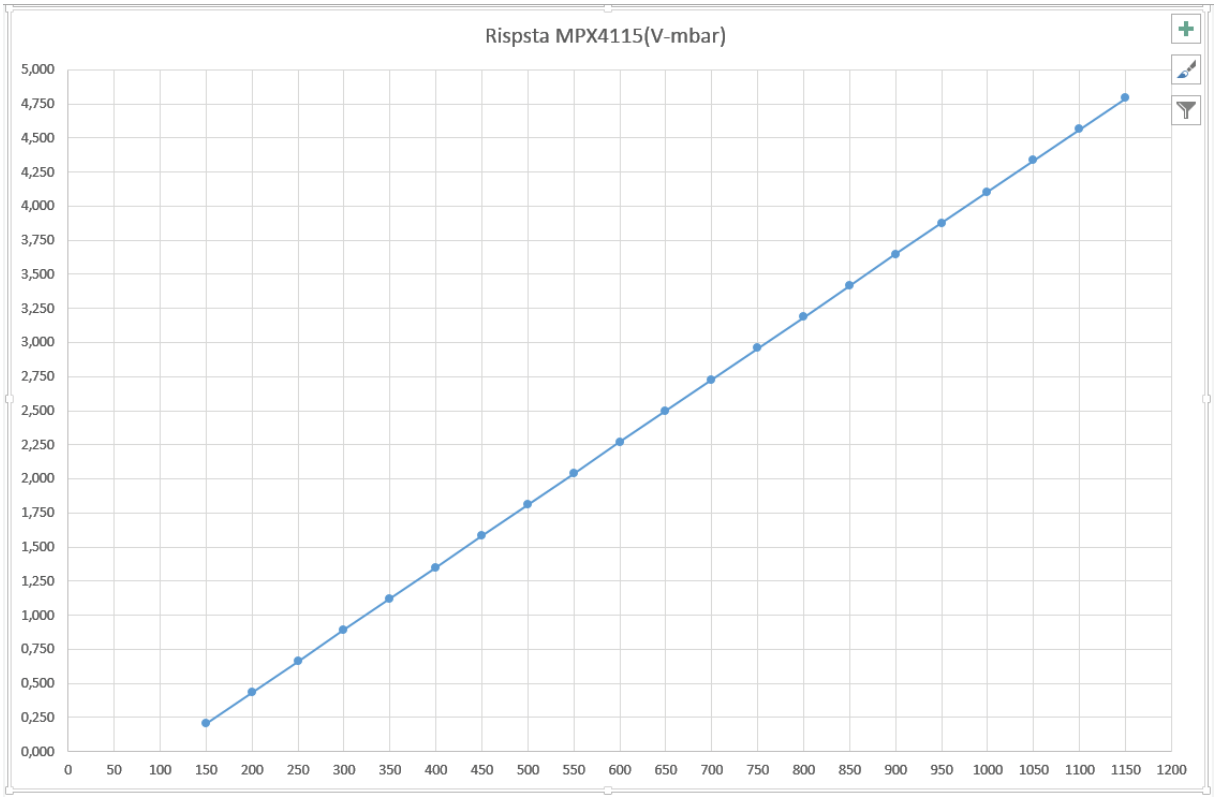


Figura 6: Risposta del sensore MPX4115(V-mbar)

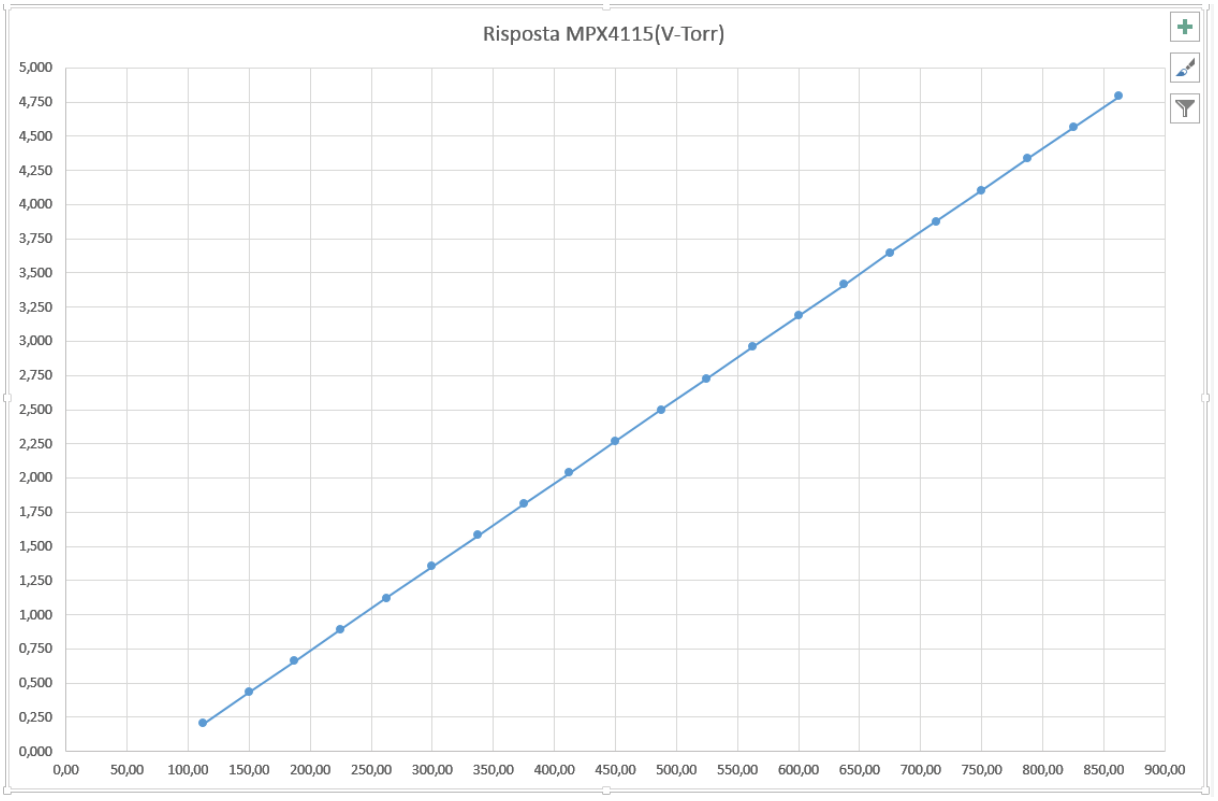


Figura 7: Risposta del sensore MPX4115(V-Torr): Il fattore di errore dovuto alla temperatura di esercizio e quello relativo alla misura della pressione sono riportati nelle figure seguenti.

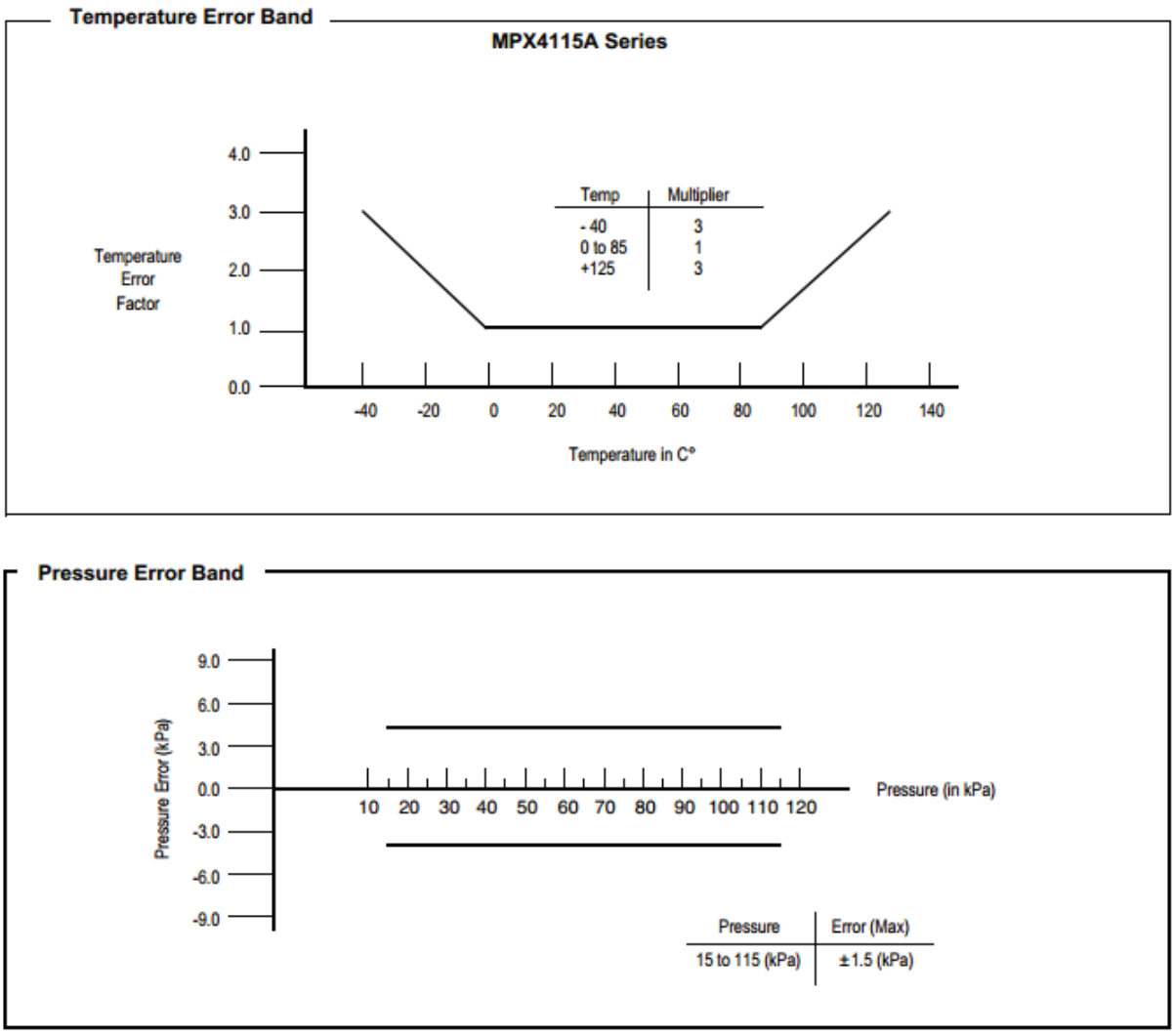


Figura 8-9: Fattore di errore temperatura e pressione

Si nota che il fattore di errore dovuto alla temperatura resta constante nell'intervallo di temperatura 0÷85 C°, per crescere linearmente, con la stessa pendenza, al disotto dello 0 C° e al disopra dei 85 C°. Questo dimostra che l'errore minimo è commesso dal dispositivo nell'intervallo di temperatura 0÷85C°,che deve essere l'intervallo di operatività per il sensore.

Il Firmware di interfacciamento

Dal momento che userò l'oscillatore interno del PIC18F45K22 è necessario impostare in modo corretto i fuses del microcontrollore,per far ciò, usando il compilatore MikroC,troveremo queste impostazioni nel pannello Edit Project.La corretta impostazione è riportata nella figura seguente.

Impostazione fuses

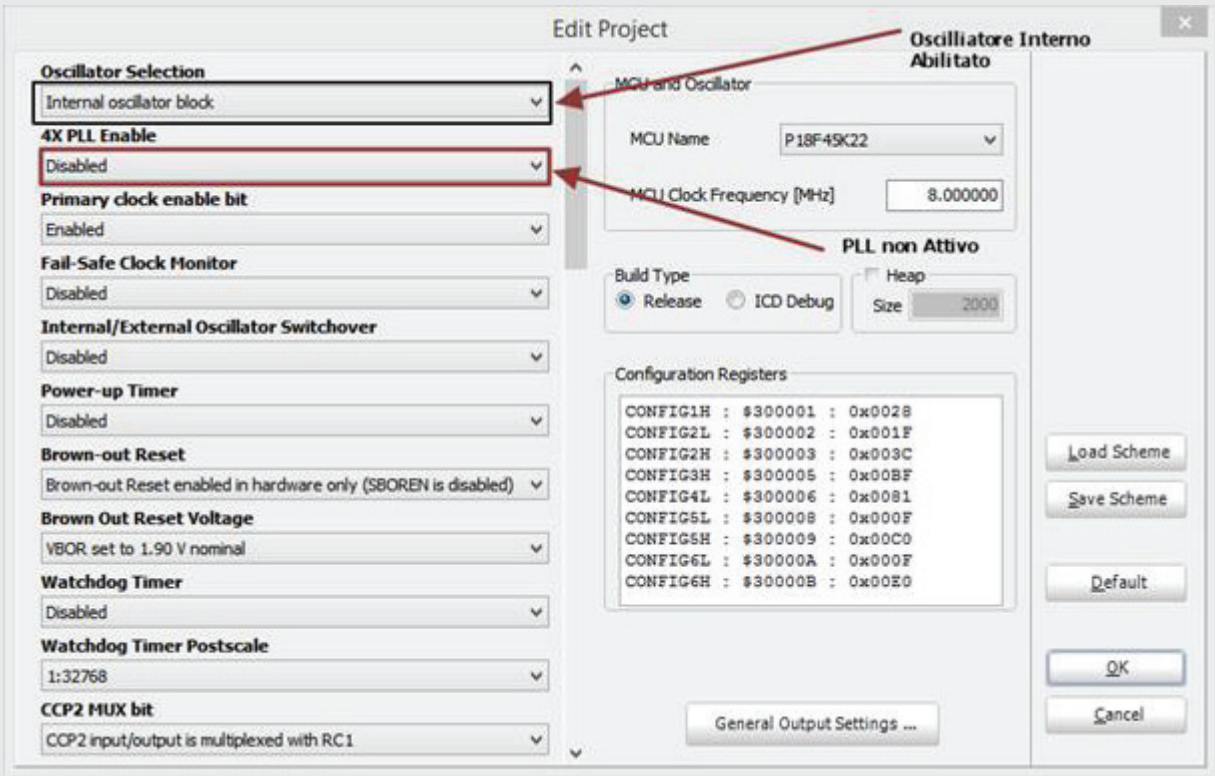


Figura 10:impostazione fuses oscillatore interno

Le impostazioni per utilizzare l'oscillatore interno devono essere inserite anche nel programma principale, ed in modo particolare subito dopo la dichiarazione della funzione main come riporta la figura sottostante.

Impostazione oscillatore interno

```
void main()
{
// Settaggio oscillatore interno a 8mhz
 OSCCON = 0b01100000;  // bit7: device enters SLEEP on sleep instruction[0]
                       // bit6-4: HFINTOSC 16MHz [111]  8MHz[110]
                       // bit3: status bit [0]
                       // bit2: status bit [0]
                       // bit1-0: clock defined by CONFIG bits [00]
// Abilitazione del PLL x4 Fosc vale a dire 64Mhz  Fosc
 OSCTUNE = 0b10000000; // bit7:  device clock derived from the MFINTOSC or HFINTOSC source
                       // bit6: PLL enabled [1]
```

Figura 11:oscillatore interno

Come ingresso analogico userò il pin1 della Porta A, configurando il convertitore ADC in modo che le sue tensioni di riferimento siano coincidenti con quelle di alimentazione del sistema. L'allineamento dei dati di conversione avverrà partendo da destra e con un tempo di acquisizione di 8 Tad(periodo del clock del convertitore),tale tempo è riferito al Tracking

and Hold interno del PIC.Per quanto riguarda la frequenza di lavoro dell' ADC questa dovrà risultare essere la frequenza dell'oscillatore interno divisa per 32.Come mostra il listato 1.

Listato 1

```
ANSELA.B1=1;          // ingresso analogico per MPX4115
TRISA.B1 =1;          //------------------------------
//------- configurazione del convertitore ADC------------
ADCON0.GO=0;       // no start conversion
ADCON0.ADON=1;     // attivo convertitore ADC

ADCON1.B3=0;       // Vref-=VDD
ADCON1.B2=0;       //
ADCON1.B1=0;       // Vref-=VSS
ADCON1.B0=0;       //

ADCON2.ADFM =1;    // allineamento a destra

ADCON2.ACQT2=1;    // tempo di acquisizione Sample/Hold
ADCON2.ACQT1=0;    // 8 TAD
ADCON2.ACQT0=0;

ADCON2.ADCS2=0;    // fattore di divisione clock
ADCON2.ADCS1=1;    // Fosc/32
ADCON2.ADCS0=0;
```

Le funzioni utilizzate per la lettura e la visualizzazione del dato relativo alla pressione(in Torr) misurata sono due, la prima(Conversione_Adc()) permette di selezionare il canale di acquisizione,mentre la seconda(Legge_Pressione()) calcola la pressione e consente di visualizzare il dato su Lcd. In quest'ultima funzione è presente nel suo interno la funzione printfloat_lcd() che permette la visualizzazione della pressione sull' Lcd,tale funzione è una libreria personale aggiunta al compilatore ma basterà modificare il programma per adattarlo alle librerie standard del compilatore.Vedi listato 2,3,4.

Listato 2

```
void Conversione_Adc(unsigned char canale);
void Legge_Pressione(unsigned char Radc_var, float Press_var,
                     unsigned int Press_Int_var,
                     unsigned char Lcd_Text_vect);
```

Listato 3

```
void Conversione_Adc(unsigned char canale)
 {
   ADCON0.CHS4=canale.B4;
   ADCON0.CHS3=canale.B3;
   ADCON0.CHS2=canale.B2;
   ADCON0.CHS1=canale.B1;
   ADCON0.CHS0=canale.B0;
 }
```

Listato 4

```
void Legge_Pressione(unsigned char Radc_var, float Press_var,
                     unsigned int Press_Int_var,
                     unsigned char Lcd_Text_vect)
{
   Go_bit=1;
   while(Go_bit==1)
    {
     asm{
          nop;
        }
    }
   Radc=((ADRESH<<8)+ADRESL);
   Press=((7.5/0.009)*(((4.88e-03*Radc)/5.0)+0.095))-10;
   floatToStr(Press,buffer);
   printfloat_lcd(3,16,5,buffer);
   Press_Int=Press;
 }
```

Lo schema elettrico del progetto, così come la simulazione, sono stati realizzati utilizzando il programma professionale Proteus 7.10.I risultati così ottenuti sono stati conformi alla verifica sperimentale usando la EasyPic7.Osservando in fase di simulazione le tensioni misurate dal voltmetro collegato in uscita dal sensore,si sono potuti incrociare i dati calcolati facendo uso del foglio elettronico EXCEL, ben visibili in figura 4.

Proteus 7.10 SP0

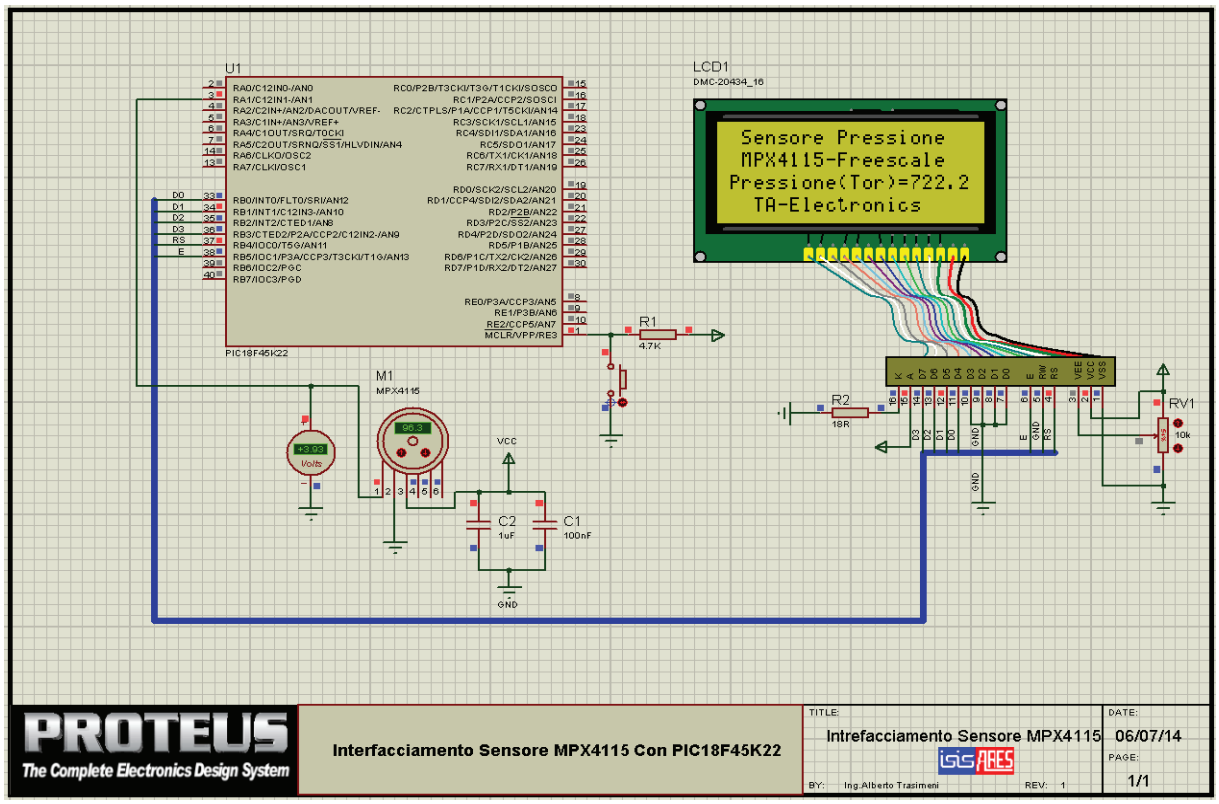


Figura 13: Simulazione del progetto

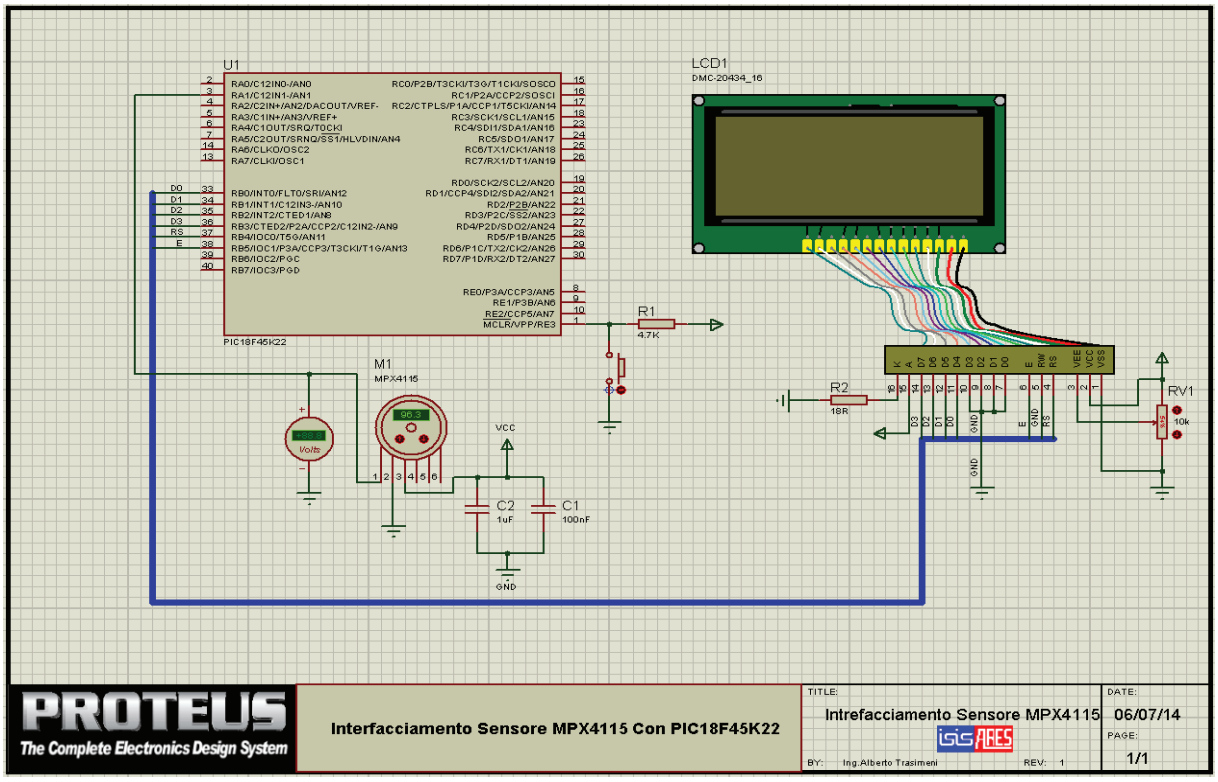


Figura 12: Schema elettrico del progetto

IL firmware del progetto, così come i file, del foglio elettronico Excel e della simulazione mediante Proteus 7.10, sono direttamente scaricabili dal sito dell'editore.
Una volta scaricati i file,per poter compilare di nuovo il programma sarà necessario modificarlo riscrivendo le seguenti funzioni, non presenti nelle librerie standard del compilatore.

Lcd_Stringa(……);
Printfloat_lcd(……..);

Se qualche lettore trovasse difficile fare questo lavoro di riscrittura, sarò ben lieto di metterle a sua disposizione, previo avvisare l'editore, che in questo caso farà da tramite.

La foto sotto riportata mostra il funzionamento del sensore MPX4115AP inserito in un contesto di una semplice stazione metereologica, che trasmette i dati via infrarosso utilizzando la modulazione OOK, oggetto di un successivo articolo.La pressione visualizzata, di 721.7 Torr, è legata ad una giornata non molto serena e all'altitudine in cui la misura è stata effettuata, di circa 300 m sul livello del mare. Ricordo che il sensore fornisce una misura di pressione assoluta.

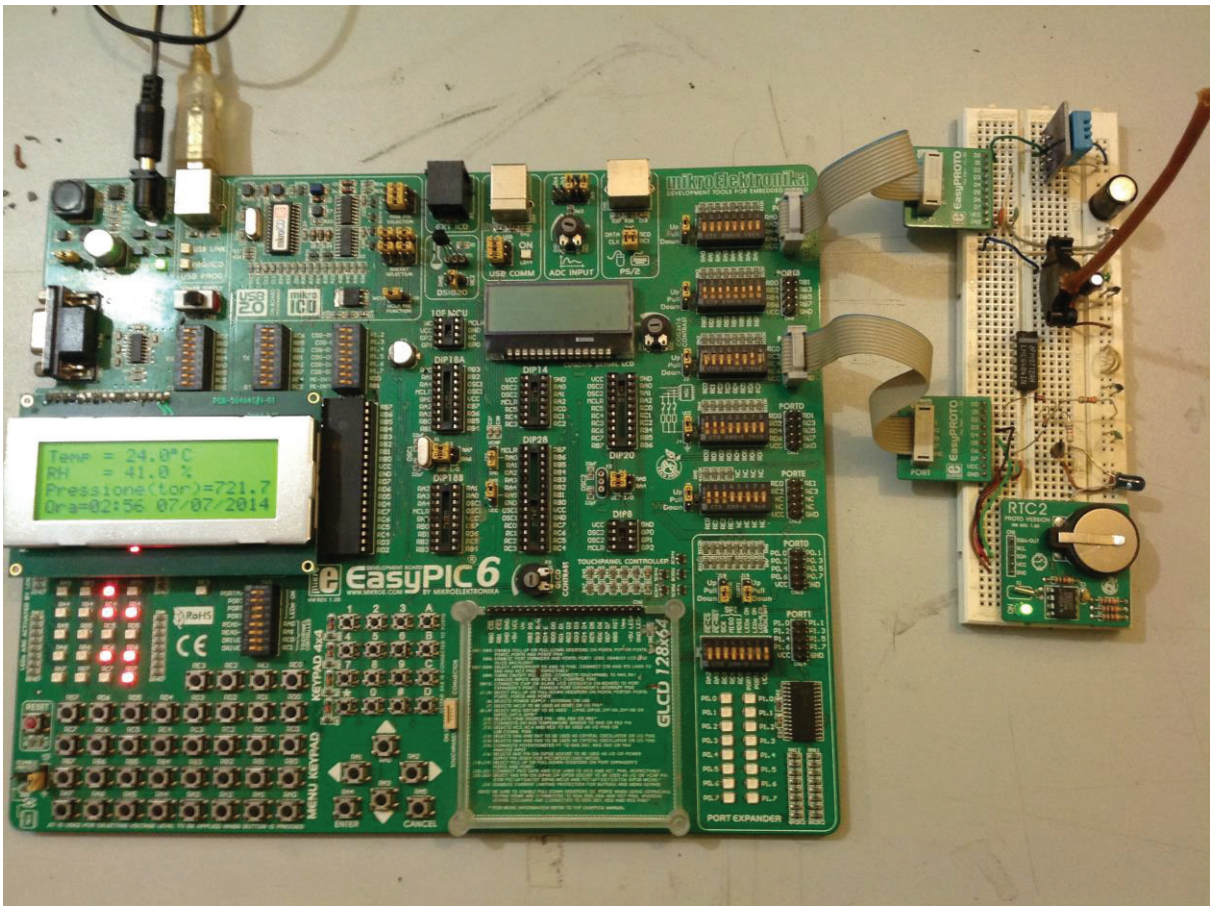


Immagine 1: Semplice stazione meteo

Osservazioni finali:

In commercio è presente l'integrato LPS331AP della STMicroelectronics che è un sensore di presssione piezoresistivo,in grado di misurare una pressione assoluta(260÷1260mbar) e risulta essere di tipo digitale. Possiede due protocolli di comunicazione ed esattamente, SPI e I2C,selezionabili via hardware mediante ponticelli.IL sensore può essere letto sia in polling che in interrupt in quanto dotato della linea INT(interruzione).La tensione massima di alimentazione e di 3.3V.
Utilizzando EasyPIC7, la Mikroelektronica fornisce il sensore già montato e sfrutta il formato mikroBus presente sulla scheda, prendendo il nome di:"Pressure click",ma può essere montato in modo stand alone su una qualsiasi board .

Schema elettrico pressure click riferito al mikroBus

Pressure Click™ Board Schematic

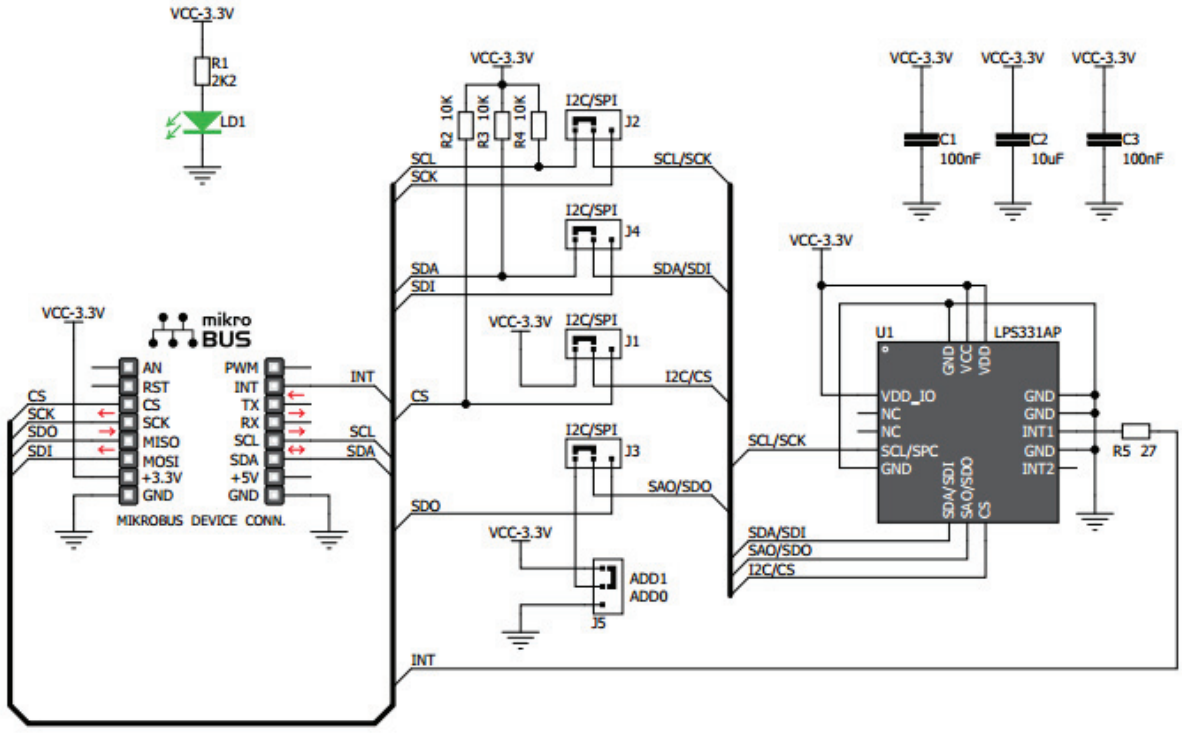


Figura 14: Pressure click

Infine riporto le dimensioni del sensore MPX4115AP con relativi pin di collegamento.

Dimensioni e pin di collegamento

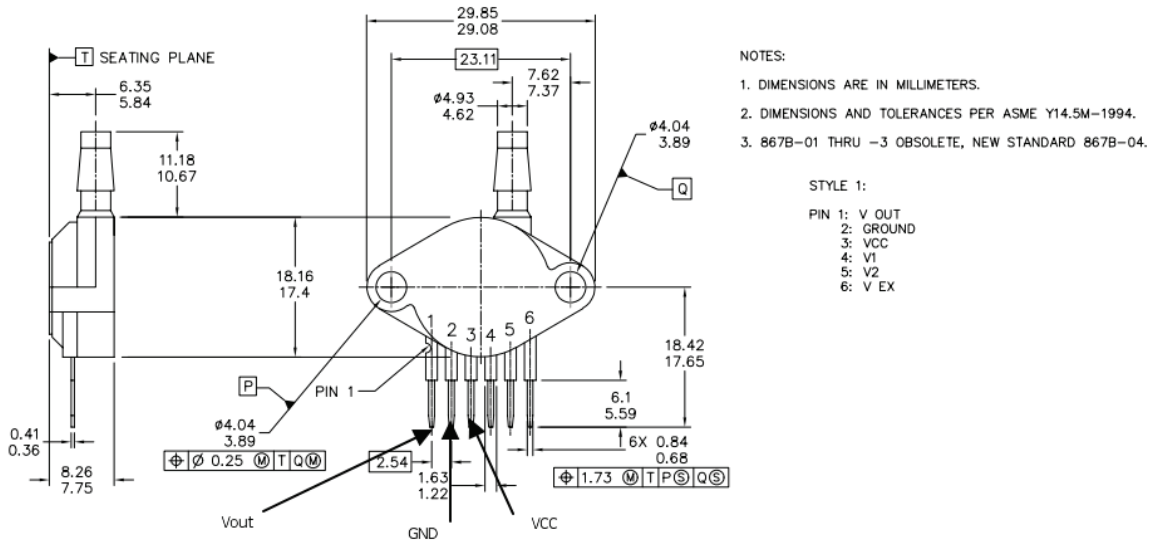


Figura 15:Dimensioni e pin

Elenco componenti	
C1	100nf multistrato
C2	1uF 25 V Tantalio
U1	PIC18F45K22
LCD	20x4 Hitachi
Sensore	MPX4115AP

Sensori di pressione e umidità

Scopo di questo articolo è quello di offrire una panoramica sui sensori di pressione ed umidità, largamente utilizzati in svariati tipi di applicazioni nel settore dell'automazione industriale, del controllo di processo, nella domotica, nel settore automotive e nell'elettronica di consumo

Introduzione

I sensori di pressione ed umidità rivestono un ruolo importante in molte applicazioni con le quali interagiamo quotidianamente. La misura di pressione è infatti richiesta, ad esempio, per rilevare la presenza di fluidi, liquidi, e gas all'interno dei circuiti idraulici, per rilevare l'altitudine, per determinare la portata di un fluido, per realizzare sistemi TPMS (Tire Pressure Monitoring System) sulle autovetture, per sistemi di misura nel campo biomedicale, ecc. I sensori di umidità hanno invece quali potenziali applicazioni le seguenti: impianti di refrigerazione e asciugatura, stazioni meteorologiche, sistemi di condizionamento ed umidificazione, domotica, apparecchiature medicali.

La misura della pressione

Anzitutto occorre definire i tre tipi fondamentali di misura della pressione, dato che i datasheet dei vari sensori disponibili sul mercato fanno riferimento a questo tipo di teminologia:

- pressione assoluta: è la pressione misurata assumendo come riferimento il vuoto, quindi la pressione nulla. Un esempio di pressione assoluta è fornito dalla pressione atmosferica, mentre un esempio di unità di misura utilizzata per questa pressione è il PaA (Pascal Absolute)
- pressione relativa (indicata in inglese con la notazione gauge pressure): è la pressione misurata rispetto alla pressione atmosferica. Un esempio di pressione relativa è rappresentato dalla pressione sanguigna, mentre un esempio di unità di misura utilizzata per questa pressione è il PaG (Pascal Gauge)
- pressione differenziale: è la pressione misurata rispetto ad una pressione di riferimento. Si può pertanto osservare come la pressione relativa sia un caso particolare di pressione differenziale. Un'unità di misura utilizzata per questa pressione è ad esempio il PaD (Pascal Differential)

I tre tipi di misura della pressione sono rappresentati in figura 1.

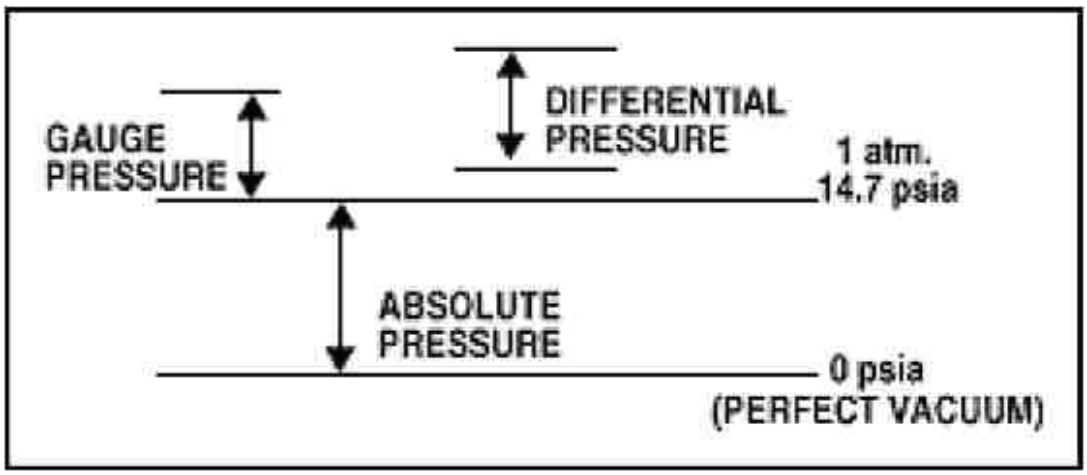


Figura 1: i tre tipi di misura della pressione

Tipologie di sensori di pressione

I sensori di pressione si prestano per essere impiegati in molteplici tipi di applicazioni nel campo dell'industria, dell'automazione, nel settore automotive, nelle apparecchiature biomedicali, e nell'elettronica di consumo in generale. I sensori di pressione integrati si possono classificare in tre tipologie, tutte con la caratteristica comune di possedere una membrana che si deforma per effetto della pressione su di essa esercitata:

- sensori di pressione piezoresistivi, basati sulla variazione di resistività di un materiale conduttore prodotta dalla deformazione della membrana
- sensori di pressione capacitivi, basati sulla variazione di capacità prodotta dal movimento della membrana
- sensori di pressione risonanti, basati sulla variazione della frequenza di risonanza di un materiale posto sulla membrana, prodotta come effetto della deformazione della stessa

Sensori di pressione piezoresistivi

Sono caratterizzati dal possedere un'uscita che varia linearmente con la pressione applicata, da un'elevata impedenza di uscita, da un costo e da una complessità di realizzazione relativamente bassi. Per contro, presentano lo svantaggio di essere poco sensibili, quindi non particolarmente adatti alla misura di basse pressioni, oltre alla dipendenza della misura dalla temperatura (per cui possono richiedere, se necessario, un'azione di compensazione della stessa). Questi sensori utilizzano in genere dei resistori realizzati in silicio monocristallino, particolarmenti adatti per realizzare dispositivi ad elevata integrazione, e caratterizzati da un elevato fattore di misura e alta sensibilità. I resistori sono solitamente connessi in configurazione a ponte di Wheatstone (linearità dell'uscita) ma, come già evidenziato, esiste una dipendenza della loro resistività dalla temperatura. Le tecnologie elettroniche di fabbricazione adottate per la loro costruzione prevedono il micromachining, il surface blonding, e il surface-micromachining. Inoltre, questi sensori sono adatti a misurare variazioni di pressione, piuttosto che pressioni statiche. I sensori piezoresistivi sono adatti alle misure di pressione assoluta, relativa, e differenziale,

Come esempio di sensore di pressione piezoresistivo prendiamo la serie MPX2010 di Freescale, capace di misurare pressioni fino a 75kPa (100kPa in modalità burst), con compensazione della temperatura e calibrazione dell'offset. La serie MPX2010 comprende sensori piezoresistivi realizzati in silicio in grado di fornire una tensione di uscita accurata e lineare, direttamente proporzionale alla pressione applicata. I sensori ospitano al loro interno un singolo blocco monolitico di silicio con integrata una sottile rete di resistori e capace di misurare le deformazioni (tensioni meccaniche) prodotte. I sensori sono inoltre regolati tramite laser per offrire una lunga durata, calibrazione dell'offset, e compensazione della temperatura nel range compreso tra 0°C e +85°C. I sensori della serie MPX2010 sono adatti alla misura di pressioni differenziali e relative, con applicazioni nel campo della diagnosi medica (soprattutto per le patologie di tipo respiratorio), nel controllo del movimento del flusso d'aria, nei sensori di pressione in generale. In figura 2 è mostrata la curva caratteristica di questo sensore; sull'asse orizzontale è riportata la pressione applicata al sensore, mentre sull'asse verticale la tensione in uscita dallo stesso. La curva è riferita alla temperatura di riferimento di 25°C. Si notino sia la linearità perfetta della curva, che il valore di offset, molto contenuto. In figura 3 è mostrato un esempio di sensore MPX2010 (esitono molte versioni con contenitori differenti): i quattro pin corrispondono alla tensioni di uscita (+VOUT e -VOUT), alla tensione di alimentazione VS e alla massa. Questo sensore è dotato inoltre di doppia porta per la misura della pressione: una corrisponde alla pressione positiva e l'altra alla pressione di riferimento del vuoto.

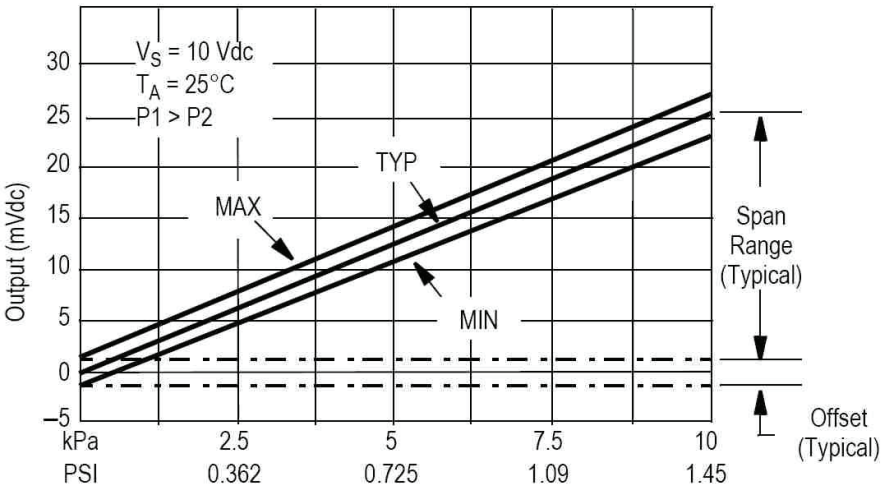


Figura 2: linearità di un sensore piezoresistivo

Figura 3: un sensore della famiglia MPX2010

Sensori di pressione capacitivi

Rispetto al precedente tipo, i sensori capacitivi sono caratterizzati da una maggiore sensibilità, una minore dipendenza della misura dalla temperatura, e da un assorbimento ridotto. Tra gli svantaggi vanno menzionati la non linearità dell'uscita e la presenza di capacità parassite che possono influenzare negativamente la misura. Il condensatore ha un'armatura fissa, solidale con il substrato del materiale adottato per la sua fabbricazione, e l'altra, mobile, solidale con la membrana; le deformazioni subite da quest'ultima generano pertanto delle variazioni di capacità sulle armature del condensatore. Le tecnologie elettroniche impiegate per la loro fabbricazione sono il fusion-bonding e il surface-micromachining. Questi sensori richiedono un circuito elettronico addizionale in grado di leggere il valore di capacità e ricavare da questo la misura di pressione corrispondente. Nei casi più semplici può essere un circuito a componenti discreti, fino ad arrivare ad un vero e proprio processore dedicato. Il valore di capacità è compreso tra pochi picofarad fino a circa 50-100 pF. Questi sensori sono particolarmente adatti alla misura di piccole pressioni, sia assolute che differenziali e relative.
Come esempio di sensore di pressione capacitivo si consideri il modello KP106 prodotto da Infineon, visibile in figura 4. Il sensore è stato progettato per impieghi automotive, in particolare per rilevare gli urti laterali e pertanto viene installato in un modulo all'interno delle portiere laterali. Quando la portiera si comprime a seguito di un impatto laterale, il sensore KP106 genera un segnale con ampiezza proporzionale alla variazione di pressione che si genera all'interno della portiera. Il segnale di uscita, indipendente dalla pressione ambiente, viene poi trasmesso alla ECU per attivare il relativo airbag.

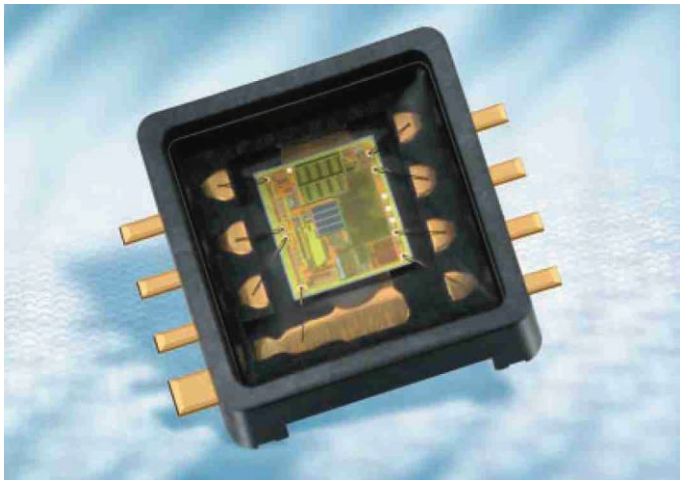


Figura 4: il sensore KP106 di Infineon

Sensori di pressione risonanti

Questa tipologia di sensori è contraddistinta da un'elevata risoluzione e precisione della misura, con un'uscita funzione dalla frequenza di risonanza. Lo svantaggio principale deriva essenzialmente dalla complessità realizzativa del sensore, che incide negativamente anche sul loro costo. Vanno inoltre considerate la non linearità del sensore e la sensibilità alla temperatura e alle vibrazioni. Per ricavare il valore di pressione partendo dalla frequenza di risonanza del sensore si utilizza solitamente un circuito oscillatore elettronico. In figura 5 è mostrato un esempio di sensore di pressione risonante, l'RTC350 prodotto da General Electric. Trattasi di un sensore caratterizzato da un'elevata stabilità (100ppm per anno) e da un'elevata accuratezza (0,001%). Il range di pressioni è compreso tra 35 e 3500 mbar assoluti, mentre l'uscita è in frequenza su un'interfaccia di tipo RS232/485.

Figura 5: il sensore RTC350 di GE

La misura dell'umidità

Con il termine umidità si intende la quantità di vapore acqueo contenuta nell'aria o in altri gas. L'umidità può essere misurata in vari modi e con diverse unità di misura; le tre più comunemente usate sono: umidità assoluta, umidità relativa (RH), e dew point (punto di rugiada).

Umidità assoluta

È definita come il rapporto tra la massa di vapore acqueo ed il volume di aria o gas che lo contiene. Viene solitamente espressa in grammi per metro cubo.

Punto di rugiada

È definito come la temperatura e pressione alle quali il gas comincia a condensarsi in liquido ed è espressa in °C o °F.

Umidità relativa

È definita come il rapporto (in percentuale) della pressione parziale del vapore acqueo contenuto in un miscuglio gassoso di aria e vapore acqueo rispetto alla pressione di vapor saturo, espressa in percentuale. Un'umidità relativa del 100% indica che il miscuglio gassoso contiene la massima quantità di umidità possibile per le date condizioni di temperatura e pressione.

Tipologie di sensori di umidità

I sensori di umidità possono essere classificati in tre categorie: sensori di umidità capacitivi, sensori di umidità resistivi, e sensori di umidità termici (o sensori di umidità assoluta).

Sensori di umidità capacitivi

Sono ampiamenti utilizzati nelle applicazioni industriali, commerciali, e in meteorologia, e misurano soprattutto l'umidità relativa. Sono composti da due elettrodi conduttivi, all'interno dei quali è presente un substrato (vetro, ceramica, o silicone) sul quale è depositata una sottile pellicola di ossido metallico. La variazione incrementale di costante dielettrica di un sensore di umidità capacitivo è all'incirca direttamente proporzionale all'umidità relativa presente nell'ambiente. La variazione di capacità è dell'ordine di 0,2-0,5 pF per una variazione di RH dell'1%, mentre la capacità complessiva a 25°C e 50% RH è compresa tra 100 e 500 pF. Questi sensori hanno un basso coefficiente di temperatura, resistono alle alte temperature, e sono resistenti agli agenti tossici. I tempi di risposta sono compresi tra qualche secondo e qualche decina di secondi. Come esempio di sensore di umidità capacitivo si consideri il modello HS1101, mostrato in figura 6. Si tratta di un sensore basato su una cella capacitiva in grado di misurare l'umidità relativa con un'accuratezza pari a ±5% ed un tempo di risposta inferiore a 5 secondi. È molto semplice da interfacciare con vari modelli di microcontrollore, oppure con un semplice circuito RC.

Figura 6: il sensore HS1101

Sensori di umidità resistivi

I sensori di umidità resistivi misurano la variazione di impedenza di un materiale igroscopico, come un polimero conduttivo, sale, oppure un substrato opportunamente trattato. La variazione di impedenza è inversamente proporzionale all'umidità: quanto più

alta è l'umidità, minore è la resistenza e quindi maggiore sarà la tensione in uscita. I sensori resistivi sono solitamente composti da elettrodi realizzati con metalli nobili depositati su un substrato rivestito con un sale o con un polimero conduttivo. Il sensore assorbe il vapore acqueo che provoca una dissociazione dei gruppi ionici con un conseguente aumento della conducibilità elettrica. Il tempo di risposta per la maggiorparte dei sensori resistivi varia tra 10 e 30 secondi, mentre l'impedenza è compresa tra 1 kΩ e 100 MΩ. Come esempio di sensore di umidità di tipo resistivo consideriamo il modello SYH-1 prodotto da Samyoung (visibile in figura 7), in grado di offrire un'accuratezza pari a ±3%RH a 25°C ed un tempo di risposta inferiore a 60 secondi. In figura 8 è mostrata la curva caratteristica del sensore, dalla quale si evince come la resistenza, espressa in kΩ, sia inversamente proporzionale all'umidità relativa. Un pregio di questo sensore è quello di presentare una curva pressochè lineare, cosa che agevola la misura dell'umidità relativa eseguita tramite microcontrollore e firmware opportuno.

Figura 7: il sensore SYH-1

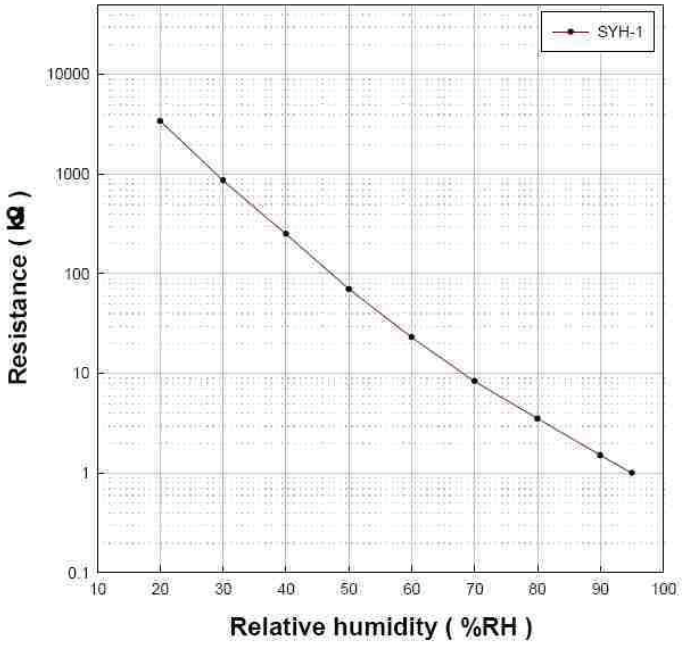


Figura 8: caratteristica del sensore SYH-1

Sensori di umidità assoluta

Questi tipi di sensori misurano l'umidità assoluta quantificando la differenza tra la conduttività termica dell'aria secca e quella dell'aria contenente vapore acqueo. I sensori di questo tipo sono in grado di lavorare anche ad elevate temperature e possono pertanto essere utilizzati in applicazioni dove i sensori capacitivi e resistivi non sopravvivrebbero.

Figura 9: il sensore ABS-FS11

L'accuratezza è dell'ordine di +3g/m3, corrispondente a circa ±5%RH a 40°C e ±0,5%RH a 100°C. Come esempio di sensore di umidità assoluto si consideri il modello ABS-FS11 prodotto da Hygrosens Instruments (figura 9), un sensore in grado di lavorare fino a 200°C ed estremamente robusto. È particolarmente indicato per applicazioni quali processi di asciugatura industriali, forni, condizionatori, analisi dei gas, forni a microonde. Il tempo di risposta è inferiore a 16 secondi, mentre il range di umidità assoluta misurabile è compreso tra 0 e 130 g/m3. Il sensore è composto da due NTC identiche, connesse in serie, attraverso le quali viene fatta passare una corrente tale per cui le NTC raggiungono una temperatura di 300°C per autoriscaldamento. Un NTC è posta in una cella contenente il gas di riferimento (azoto, corrispondente alla condizione di aria secca), mentre l'altra NTC è posta in una cella collegata con l'ambiente esterno. A seconda della conduttività termica del gas da misurare, si genererà una differenza di potenziale tra le due celle e questa è una misura della concentrazione del gas rispetto al gas di riferimento.

CONTROLLO ACCESSI CON PIC

di Giovanni Di Maria

Il controllo degli accessi ai dispositivi è oggi un'operazione molto semplice grazie alla tecnlogia messa a disposizione da MikroElektronica.

RFid Reader è una scheda che ha lo scopo di leggere le Identification Cards (RFid Cards) con il supporto delle onde radio. Tale scheda funge da modulo ricevente e trasmittente tramite antenna, ed è corredata da un connettore maschio 2x5 per permetterne la connessione ad un sistema di sviluppo.

Cos'è RFid

Radio frequency identification (RFID) è una tecnologia di un sistema capace di trasmettere l'identità (solitamente un numero seriale univoco) di una persona, animale o oggetto, usando le onde radio.
Tale tecnologia è molto utilizzata. Si pensi, ad esempio, ad un cane provvisto di chip, per il suo riconoscimento e tracciabilità.
In più tale tecnologia è resa ancor più sicura, adottando anche metodologie e applicazioni biometriche.
Diversamente dai lettori a codici a barre, questa tecnologia non richiede un contatto fisico e diretto. I dati possono anche essere letti attraverso il corpo umano ed altri tipi di ostacoli.

Componenti RFid

Ecco a grandi linee i componenti principali di un sistema RFid:

- L'**antenna**, che emette onde radio, attivando il trasponder. Quindi legge ed invia informazioni a quest'ultimo;
- Il **reader**, che trasmette i segnali anche a parecchi metri, secondo la potenza irradiata;
- Il **tag** o **trasponder**, che nelle vicinanze del reader si attiva.

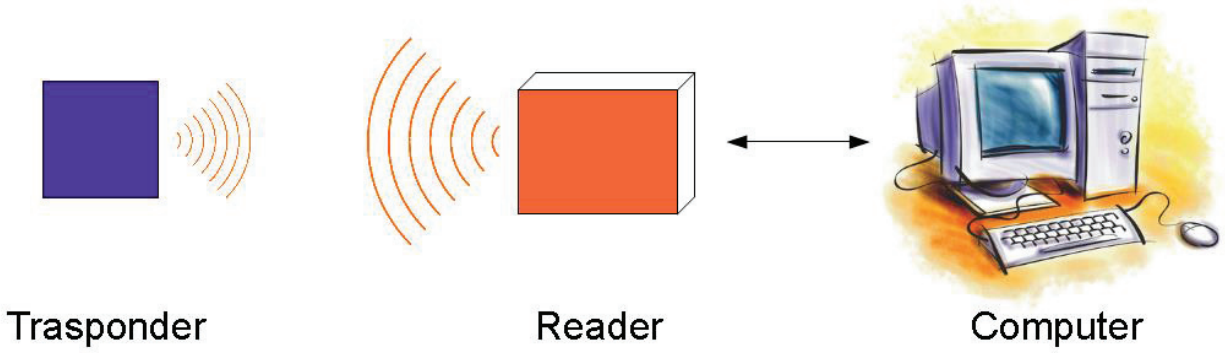


Figura 1: Principio della tecnologia RFid

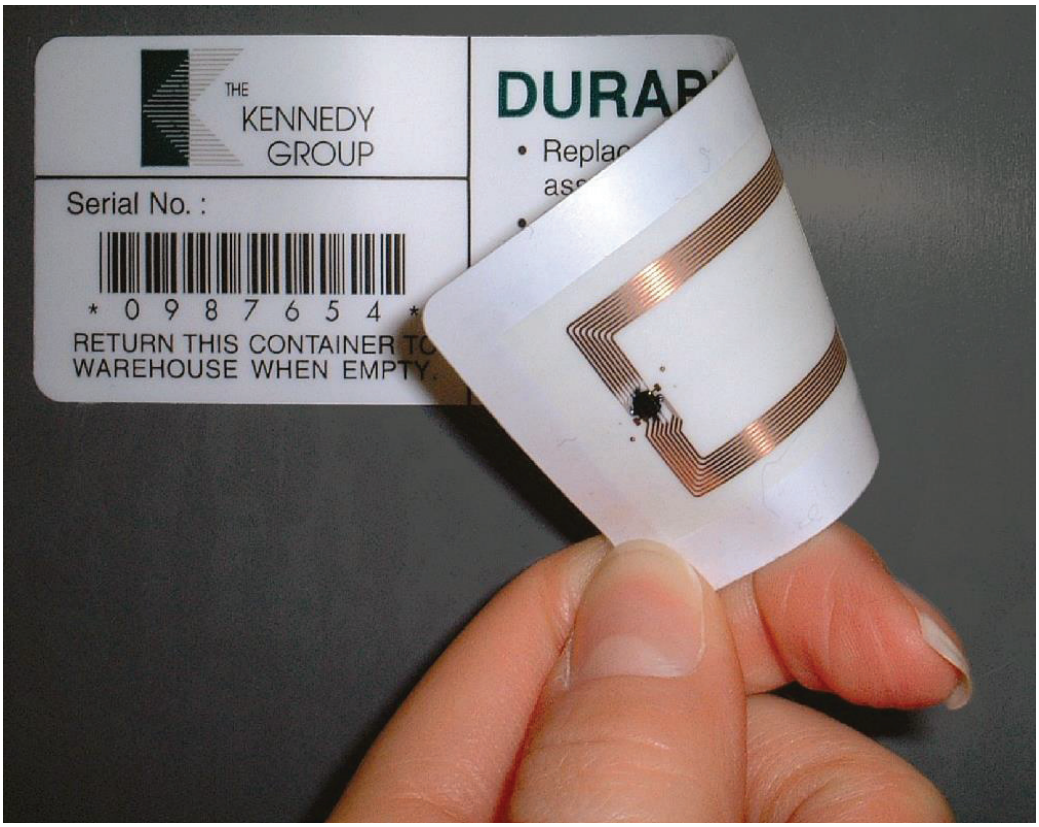


Figura 2: Esempio di utilizzo RFid

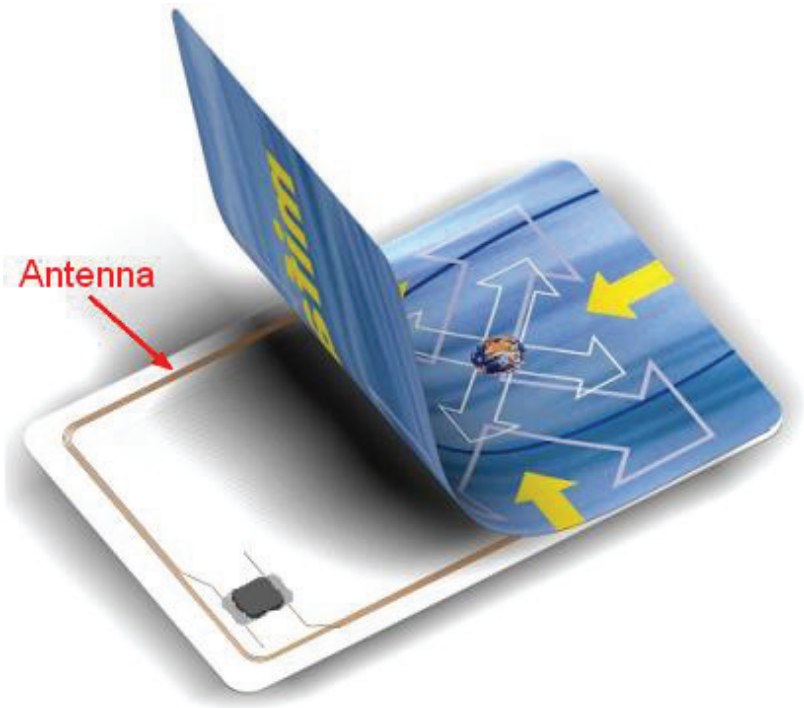


Figura 3: Smart card RFid

Applicazioni

Molteplici sono le applicazioni in cui si utilizza la tecnologia RFid, ed il numero di esse incrementa sempre più, con il passare del tempo. A grandi linee ecco i settori di principale

utilizzo:

- Produzione
- Vendita al dettaglio
- Sistemi di pagamento
- Controllo e sicurezza
- Controllo accessi
- Trasporti
- Carte di credito
- Controllo presenze
- Assistenza e manutenzione
- Identificazione degli animali
- Biblioteche
- Antitaccheggio
- Registro Scolastico Elettronico
- Monitoraggio raccolta rifiuti
- Bigliettazione Elettronica
- Logistica Magazzini
- E tanti altri...

RFid Reader Board della MikroElektronica

RFid Reader è una scheda prodotta dalla MikroElektronika, utilizzata per leggere le Identification Cards (RFid Cards) tramite onde radio. Essa funziona da ricevente e trasmittente tramite antenna. Puà essere connessa ad un sistema di sviluppo (tipo Easy Pic 5) per mezzo di un connettore maschio 2x5.
Il suo funzionamento si basa su due princìpi fondamentali:

- L'amplificazione della modulazione delle onde radio;
- L'induzione elettromagnetica.

In dettaglio

RFid Reader è costituita dalle seguenti componenti:

- Un connettore maschio (2x5) che permette il collegamento ad una scheda di sviluppo (tipo Easy Pic 5). I collegamenti prevedono i segnali di alimentazione (Vcc e Gnd) e naturalmente l'accesso al bus delle porte di un pic (ad esempio: RB0, RB1, RB2, RB3, RB4, RB5, RB6, RB7);
- Il Transceiver EM4095 che è utilizzato quale modulatore e demodulatore AM e anche come driver per antenna;
- L'antenna vera e propria, “stampata” sulla scheda, che riceve i segnali provenienti dalla RFid Card.

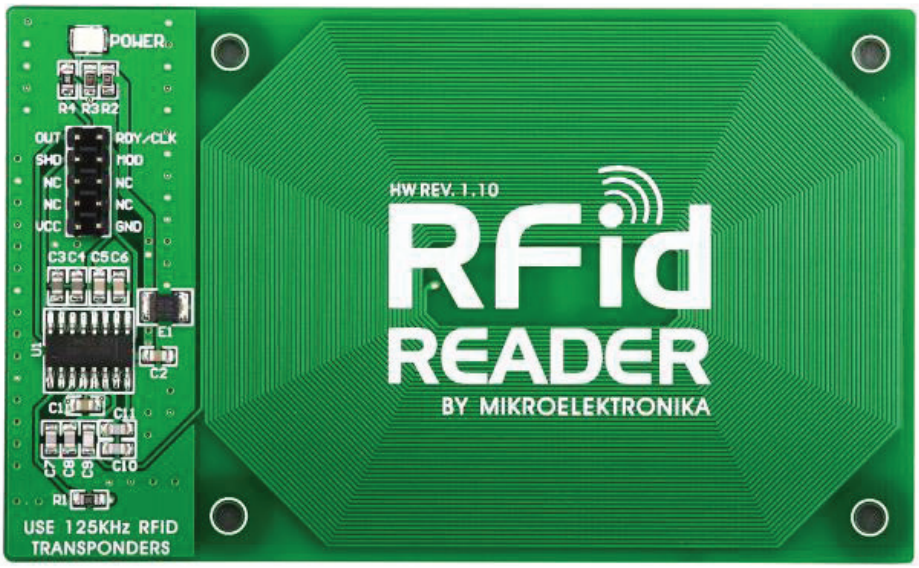


Figura 4: Il lettore RFid

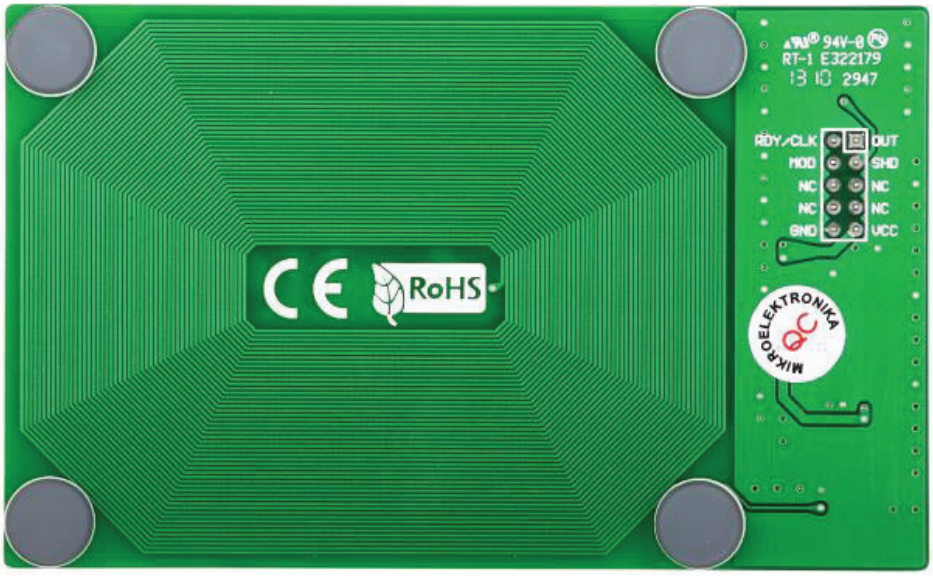


Figura 5: Il lettore Rfid (vista posteriore)

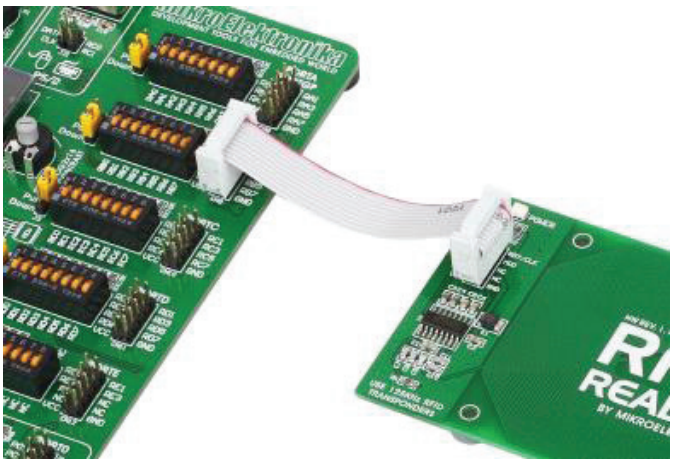


Figura 6: La connessione ad un sistema di sviluppo

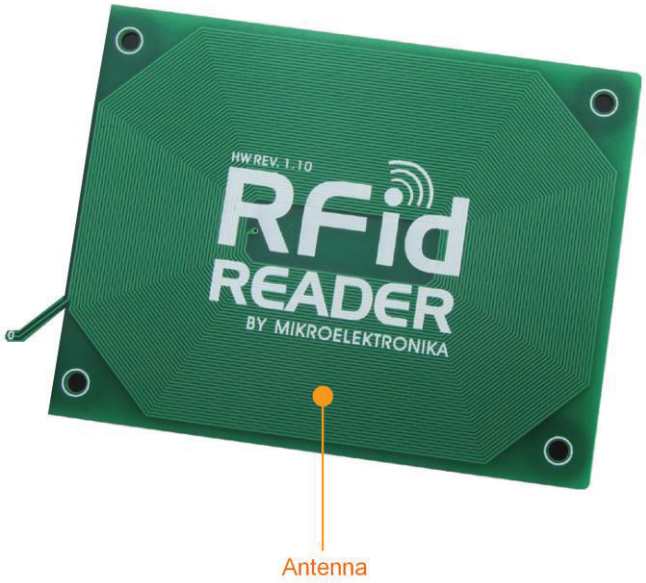


Figura 7: L'antenna integrata

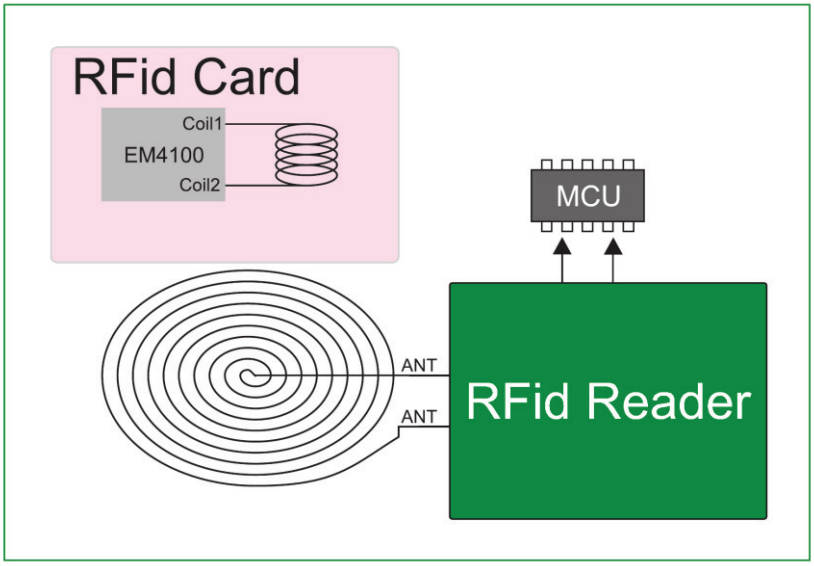


Figura 8: Schema di principio

RFid Reader è alimentato quando il sistema di sviluppo (ad es: Easy Pic 5) è collegato ad esso. La presenza di alimentazione sul Reader è indicata da un diodo led (Power). Non appena il modulo viene acceso, l'antenna riceve una tensione alla frequenza di 125 Khz ed emette un campo elettromagnetico atto a leggere le Identification Cards.

Algoritmi di sicurezza

RFid Reader dispone di alcuni sofisticati algoritmi per evitare problemi o errori nella lettura dei dati a distanza. In particolare:

- Introduce un sistema logico algoritmico di Anti Collisione, grazie al quale è possibile la lettura simultanea anche di un numero elevato di Identification Cards, il tutto privo di qualsiasi tipo di errore;
- Include anche un sofisticato sistema per cui un Tag (o Trasponder) viene letto solamente una vola.

Identification Cards

Le Identification Cards non posseggono una alimentazione elettrica propria ma non appena si avvicinano al Reader, esse vengono alimentate grazie al campo magnetico indotto. Il chip on-board contiene un codice univoco identificativo, che viene inviato non appena la Card si avvicina all'unità di lettura. Quindi la stessa unità riceve il codice e lo invia ad un microcontrollore, per la sua successiva elaborazione.

Figura 9: Una Identification Card

Schema elettrico

Di seguito viene mostrato lo schema elettrico relativo al RFid Reader. Tutte le componenti sono estremamente miniaturizzate, e nei modelli più recenti anche i componenti passivi sono in SMD. Occupa un po' di spazio in più l'antenna stampata, per ovvie ragioni.
Ecco in breve la funzione dei pin del connettore 2x5, il quale permette la connessione ed il colloquio con un microcontrollore:

- OUT: segnale di uscita dall'integrato EM4095 che contiene, tra l'altro, il codice univoco della Identification Card;
- RDY/CLK: è il piedino che detta il segnale di clock;
- SHD: se presente un segnale logico basso su questo piedino, RFid Reader entra in modalità sleep;
- MOD: se presente un segnale logico alto su questo piedino, il segnale di modulazione viene emesso dall'antenna e la trasmissione ha luogo.

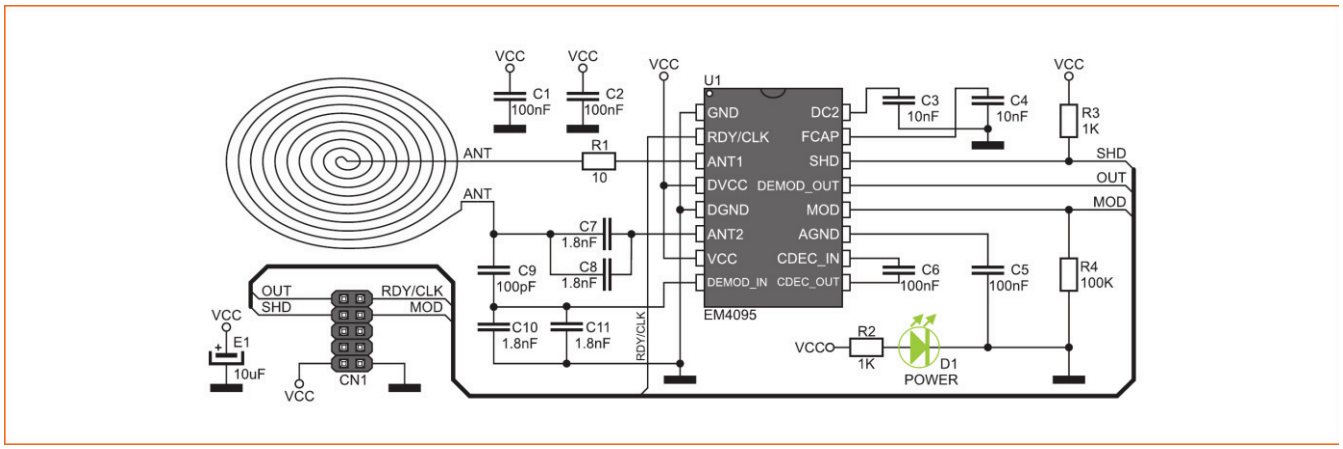


Figura 10: Schema elettrico e connessioni RFid Reader

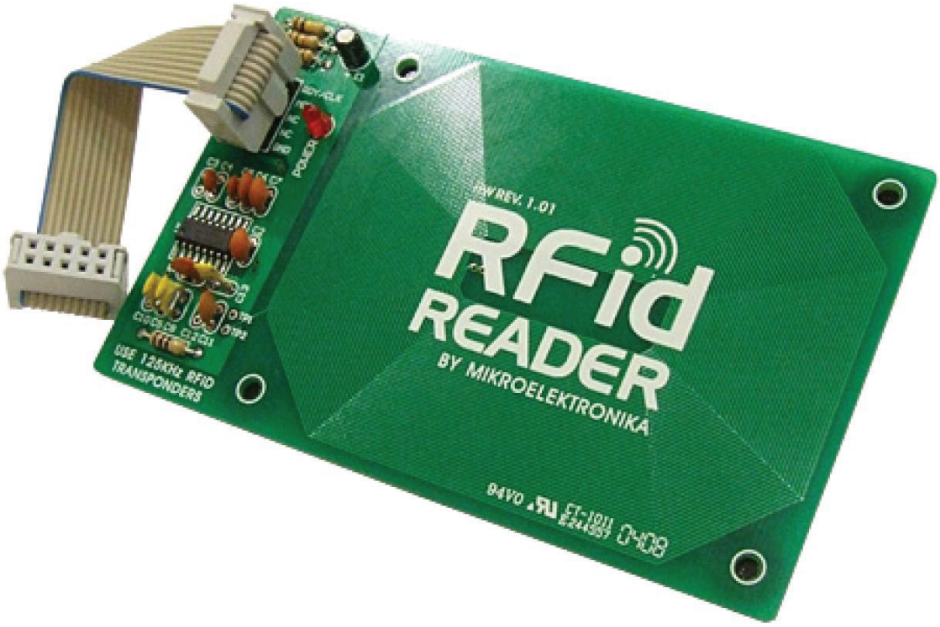


Figura 11: RFid Reader completa di cavetto per la connessione

Esempio

Di seguito è riportato un esempio di applicazione, creato con mikroBasic. Per eventuali modifiche e commenti si rimanda il lettore al sito ufficiale.

```
program RFiD

dim OUT as sbit  at RB0_bit
dim RDY_CLK as sbit at RB1_bit
dim SHD as sbit at RB2_bit
dim _MOD as sbit at RB3_bit

dim OUT_Direction as sbit at TRISB0_bit
dim RDY_CLK_Direction as sbit at TRISB1_bit
dim SHD_Direction as sbit at TRISB2_bit
dim _MOD_Direction as sbit at TRISB3_bit

dim sync_flag,
    one_seq,
    data_in,
    data_index,
    cnt,
    cnt1, cnt2 as byte
    i as byte
    _data as byte[256]
    data_valid as byte[64]

    bad_synch as byte

sub procedure Interrupt()

  if ((INT1IF_bit = 1) and (INT1IE_bit = 1)) then
      Inc(cnt)
```

```
      INT1IF_bit = 0

  else if ((INT0IF_bit=1) and (INT0IE_bit=1)) then
       cnt = 0
       sync_flag  = 1
       INT0IF_bit = 0
       INT0IE_bit = 0
       INT1IF_bit = 0
       INT1IE_bit = 1
      end if
  end if
end sub

sub function CRC_Check(dim byref bit_array as byte[64]) as byte

dim row_count, row_bit, column_count as byte
    row_sum, column_sum as byte
    row_check as byte[5]
    column_check as byte[11]

   row_count = 9
   while row_count < 59
     column_count = 0
     while column_count < 5
       row_check[column_count] = bit_array[row_count+column_count]
       Inc(column_count)
     wend
     row_bit = 0
     row_sum = 0
     while row_bit < 4
       row_sum = row_sum + row_check[row_bit]
       Inc(row_bit)
     wend

     if row_sum.B0 <> row_check[4].B0 then
       result = 0
       exit
     end if
     row_count = row_count + 5
   wend

   column_count = 9
   while column_count < 13
     row_bit = 0
     row_count = 0
     while row_bit < 11
       column_check[row_bit] = bit_array[column_count+row_count]
       Inc(row_bit)
       row_count = row_count + 5
     wend
     row_bit = 0
     column_sum = 0
     while row_bit < 10
```

```
        column_sum = column_sum + column_check[row_bit]
        Inc(row_bit)
      wend
      if column_sum.B0 <> column_check[10].B0 then
        result = 0
        exit
      end if
      Inc(column_count)
    wend

    if bit_array[63] = 1 then
      result = 0
      exit
    end if

    result = 1

end sub

'** main program

main:

  ADCON1 = 0x0F
  CMCON = 7

  OUT_Direction = 1
  RDY_CLK_Direction = 1
  SHD_Direction = 0
  _MOD_Direction = 0

  SHD = 0
  _MOD = 0

  UART1_Init(19200)
  Delay_ms(100)

  sync_flag = 0
  one_seq = 0
  data_in = 0
  data_index = 0
  cnt = 0
  cnt1 = 0
  cnt2 = 0

  INTEDG0_bit = 0
  INTEDG1_bit = 1
  INT0IF_bit = 0
  INT1IF_bit = 0

  INT0IE_bit = 0
  INT1IE_bit = 0
  GIE_bit = 1
```

```
while TRUE
   bad_synch = 0
   cnt = 0
   sync_flag = 0
   INT1IF_bit = 0
   INT1IE_bit = 0
   INT0IF_bit = 0
   INT0IE_bit = 1

   while (sync_flag = 0)
     nop
   wend
   while (cnt <> 16)
     nop
   wend
   cnt = 0
   _data[0] = OUT and 1
   data_index = 1
   while data_index <> 0
      while (cnt <> 32)
        nop
      wend
      cnt = 0
      _data[data_index] = OUT and 1
   if (data_index and 1) then
   if (_data[data_index] xor _data[data_index-1]) = 0 then
     bad_synch = 1
     break
   end if
   end if
   Inc(data_index)
   wend
   INT1IE_bit = 0

   if (bad_synch) then
     continue
   end if
   cnt1 = 0
   one_seq = 0
   for cnt1 = 0 to 127
     if (_data[cnt1 << 1] = 1) then
        Inc(one_seq)
     else
        one_seq = 0
     end if
     if (one_seq = 9) then
       break
     end if
   next cnt1
   if (one_seq = 9) and (cnt1 < 73) then

       data_valid[0] = 1
```

```
            data_valid[1] = 1
            data_valid[2] = 1
            data_valid[3] = 1
            data_valid[4] = 1
            data_valid[5] = 1
            data_valid[6] = 1
            data_valid[7] = 1
            data_valid[8] = 1
            for cnt2 = 9 to 63
               Inc(cnt1)
               data_valid[cnt2] = _data[cnt1 << 1]
            next cnt2
            if (CRC_Check(data_valid) = 1) then
               UART1_Write_Text("CRC CHECK OK!")
               UART1_Write(13)
               UART1_Write(10)
               for i = 0 to 64
                 if data_valid[i] = 0 then
                   UART1_Write("0")
                 else
                   UART1_Write("1")
                 end if
               next i
               UART1_Write(13)
               UART1_Write(10)
               Delay_ms(500)
            end if
         end if

  wend
end.
```

Conclusioni

Come si è visto, implementare un sistema di controllo è abbastanza semplice. Il microcontrollore si occupa di tutte le operazioni derivanti dal riconoscimento o meno della IC, con abbassamento di costi e di tempo di sviluppo.

Gestire le uscite di un PIC

Ecco come scrivere in C degli algoritmi dedicati alla gestione delle uscite dei PIC, per comandare LED, display ed altri dispositivi, utilizzando le tecniche più efficienti utilizzando il compilatore MikroC.

Di seguito è mostrato il codice per fare lampeggiare un LED collegato alla porta RB0 del PIC. Per comodità riportiamo in figura 1 anche lo schema elettrico del circuito:

```
void main() {
PORTB = 0;
TRISB = 0;
while(1) {
PORTB = PORTB^0b00000001;
Delay_ms(500);
}
}
```

Il programma è molto semplice: utilizza solamente un'operazione di XOR (l'operatore "^" del C) per ottenere l'inversione del primo bit della porta B (e quindi il lampeggio del LED), ed una routine di ritardo per determinarne la frequenza. Se dovessimo generare delle combinazioni più complesse (come avviene in molte applicazioni reali) questo approccio potrebbe non essere adeguato. Esistono diverse possibilità che è il caso di considerare, dal momento che ciascuna di esse può adattarsi meglio a specifiche applicazioni. Supponiamo ad esempio di utilizzare 8 LED, ciascuno collegato ad un piedino della porta B come mostrato in figura 2, e di volerli accendere in sequenza nelle due direzioni (il classico effetto "supercar"). Le alternative più convenienti sono due: in un caso possiamo utilizzare gli operatori del C che permettono di manipolare i bit, nell'altro possiamo utilizzare una tabella di consultazione (look-up table). Consideriamo il primo caso, cioè partiamo da una configurazione delle uscite, ed otteniamo le altre operando sui bit. Il codice è il seguente:

```
void main() {
char dir;
/* Direzione di scorrimento
1=sinistra, 0=destra */
dir=1;
// Inizializz. porta B
PORTB = 0x01;
TRISB = 0;
// Loop infinito
while(1) {
// Raggiunti bordi?
if (PORTB&0x01) dir=1;
if (PORTB&0x80) dir=0;
// Aggiornamento direzione
if (dir) {
PORTB = PORTB << 1;
} else {
PORTB = PORTB >> 1;
```

```
}
Delay_ms(100);
}
}
```

Il funzionamento del programma dovrebbe risultare abbastanza intuitivo: inizialmente viene dichiarata una variabile (*dir*) per tenere traccia della direzione di scorrimento, in seguito vengono inizializzate le variabili e le porte, e poi viene eseguito il loop principale. Durante il funzionamento la variabile *dir* assumerà soltanto due valori (scelti da noi), quindi è conveniente dichiararla del tipo più piccolo possibile gestito dal C, cioè **char** (8 bit). Le linee della porta B sono state configurate tutte come uscite (TRISB=0), e soltanto il primo piedino è stato posto alto (PORTB=0x01). Ricordiamo che il MikroC permette di esprimere i valori numerici anche in binario (anteponendo "0b" al numero), ma abbiamo preferito utilizzare la notazione esadecimale per rendere il codice maggiormente portabile. Per convertire questi valori in binario si può utilizzare il convertitore Qconvertor, che si trova in basso nella finestra del messaggi del MikroC. All'interno del loop principale viene dapprima verificato se il LED attualmente acceso è il primo o l'ultimo, e in ciascuno di questi due casi viene cambiata la direzione dello scorrimento. Per eseguire questo controllo è stata utilizzata l'istruzione **if** del C: solo se la condizione dentro le parentesi è verificata l'espressione che segue vene eseguita. L'espressione che viene controllata è l'AND binario tra i bit della porta B e 0x01 (oppure 0x08). Questa espressione darà 0x01 solo se il bit meno significativo della porta B sarà alto (lo stesso per il più significativo nel caso di 0x80), altrimenti zero. Per maggiore chiarezza avremmo potuto scrivere:

```
if ((PORTB&0x01)==0x01) dir=1;
if ((PORTB&0x80)==0x80) dir=0;
```

ma ricordiamo che il C considera vera qualsiasi espressione che non valga 0, quindi anche non indicando il valore atteso otteniamo lo stesso risultato. Abbiamo usato lo stesso accorgimento per il frammento di codice successivo, che è quello che si occupa dell'aggiornamento della posizione. Se la variabile *dir* vale 1 viene eseguito uno scorrimento binario di una posizione a sinistra (a destra nell'immagine), altrimenti (**else**) viene eseguito uno scorrimento a destra (a sinistra nell'immagine). Gli scorrimenti (operatori "<<" e ">>" del C) produrranno la sequenza di valori: 0x01, 0x02, 0x04, 0x08, 0x10…, 0x80, e poi a ritroso (quando cambierà la direzione di scorrimento) fino a 0x01. Dopo ogni scorrimento attendiamo 100ms usando l'apposita routine di libreria Delay_ms, messa a disposizione dal MikroC, e ripetiamo l'intera routine, all'infinito. Consideriamo adesso un altro approccio per ottenere lo stesso risultato. In questo caso utilizzeremo una look-up table per memorizzare le configurazioni delle uscite che vogliamo ottenere. Il codice è il seguente:

```
void main() {
char n=0;
char leds[14]={0x01, 0x02,
0x04, 0x08, 0x10, 0x20,
0x40, 0x80, 0x40, 0x20,
0x10, 0x08, 0x04, 0x02};
// Inizializz. porta B
TRISB = 0;
// Loop infinito
while(1) {
PORTB=leds[n];
```

```
n++;
if (n==14) n=0;
Delay_ms(100);
}
}
```

Per creare la tabella di consultazione abbiamo utilizzato un array (*leds[…]*), in cui abbiamo inserito manualmente le configurazioni delle uscite volute (che sono le stesse che abbiamo ottenuto col programma precedente). Ci occorrono soltanto 14 voci per la tabella, che sono state inizializzate direttamente nella dichiarazione dell'array. Inoltre abbiamo dichiarato un'altra variabile (*n*) che verrà utilizzata come contatore, cioè come indice per scorrere la tabella. Nel loop principale assegniamo direttamente il valore corrente della tabella alla porta B, inoltre incrementiamo l'indice n (con l'istruzione n++). Quando l'indice raggiungerà il valore 14, sarà riportato a 0, e quindi il conteggio ripartirà da questo valore. Notare la differenza tra "==" ed "=" nell'istruzione **if**. Ricordiamo che il primo è un operatore relazionale, che controlla se il valore dell'espressione di sinistra è uguale a quello dell'espressione di destra (restituendo un valore di verità). L'altro operatore invece assegna un valore ad una variabile. Questa seconda versione del programma, sebbene richieda un po' più di memoria, ha un vantaggio importante: basta cambiare i valori in tabella per ottenere comportamenti diversi. Si provi ad esempio ad utilizzare questi valori:

```
char leds[14]={0x81, 0x42,
0x24, 0x18, 0x18, 0x24,
0x42, 0x81, 0x42, 0x24,
0x18, 0x18, 0x24, 0x42};
```

Ci saranno due LED accesi che si muoveranno in direzione opposta (basta convertire in binario i valori per rendersene conto). L'approccio usato nel primo programma invece risulta più flessibile quando le configurazioni utilizzate non sono sempre le stesse, ma variano nel tempo, magari in base a condizioni esterne.

PILOTARE I DISPLAY A 7 SEGMENTI

I display a 7 segmenti rappresentano un economico e funzionale dispositivo di visualizzazione. Vedremo quindi in questo paragrafo come utilizzarli. Un display a 7 segmenti è composto da
8 LED disposti a formare 7 segmenti più un punto decimale (non sempre presente), con cui è possibile rappresentare numeri, lettere e simboli grafici. I display a 7 segmenti sono disponibili in due versioni, che differiscono per i collegamenti dei LED: in entrambi i casi i LED possono essere pilotati indipendentemente da un terminale, mentre l'altro è comune a tutti. Il terminale comune può essere l'anodo o il catodo dei LED (e da questo particolare deriva il nome del display). Quando il terminale comune è il catodo (figura 3), quest'ultimo sarà collegato a massa, e per accendere un segmento sarà sufficiente fornire un'alimentazione positiva al rispettivo anodo (ad esempio utilizzando un piedino di uscita portato a livello logico alto). Nel caso di display ad anodo comune (figura 4), questo terminale verrà collegato al positivo dell'alimentazione, e per accendere un segmento sarà necessario portare a massa il catodo del segmento che si vuole accendere (usando un piedino di uscita a livello logico basso). Per pilotare un display a 7 segmenti avremo bisogno quindi di almeno 8 uscite del PIC, e con queste potremo gestire soltanto un carattere. Per fare le prime prove colleghiamo il display al PIC come mostrato in figura 5 (stiamo quindi utilizzando un display a catodo comune): i segmenti verranno pilotati tramite i piedini della porta B. Possiamo quindi procedere come negli esempi visti fino ad ora,

l'unico problema è che dobbiamo ricavare per ogni carattere che vogliamo visualizzare il corrispondente valore da inviare alla porta B (che esprime quali segmenti sono accesi e quali spenti). Per fare questo possiamo utilizzare l'apposito strumento messo a disposizione dal MikroC, richiamabile selezionando la voce **Seven Segment Convertor** dal menu **Tools** (figura 6). È sufficiente disegnare il carattere che vogliamo visualizzare per avere il valore numerico corrispondente ad uno dei due tipi di display. Nel nostro caso ricaveremo il valore per le cifre da 0 a 9 e per le lettere da A, b, c, d, E ed F (che ci potrebbero servire per rappresentare valori esadecimali). Un modo molto semplice per associare questi caratteri ai valori numerici consiste nell'utilizzare un array. Vediamo quindi come realizzare un programma che conta continuamente da 0 a 9:

```
void main() {
char n=0;
char segs[16]={0x3F, 0x06,
0x5B, 0x4F, 0x66, 0x6D,
0x7D, 0x07, 0x7F, 0x6F,
0x77, 0x7C, 0x39, 0x5E,
0x79, 0x71};
// Inizializz. porta B
TRISB = 0;
// Loop infinito
while(1) {
PORTB=segs[n];
n++;
if (n==10) n=0;
Delay_ms(1000);
}
}
```

Come si può notare il programma è quasi identico a quello visto precedentemente, sono stati modificati soltanto i valori contenuti nell'array, che in questo caso codificano i caratteri voluti. Se si estende il conteggio fino a 16 sarà possibile visualizzare anche le cifre esadecimali da A a F. Non possiamo superare 16 perché altrimenti leggeremmo una posizione dell'array che non esiste, ottenendo un risultato imprevedibile. E se volessimo utilizzare più cifre, e quindi più unità a 7 segmenti? Occorrerebbe utilizzare molte più porte di I/O, oppure ricorrere alla tecnica del multiplexing, molto utilizzata in questi casi. Occorre collegare i display come mostrato in figura 7 ed utilizzare soltanto 4 linee di I/O aggiuntive (questa volta provenienti dalla porta A) per gestire le 4 cifre. Il funzionamento è semplice: le linee della porta B saranno condivise da tutti i display, che quindi riceveranno gli stessi dati, ma soltanto un display alla volta sarà acceso, comandando attraverso la porta A i transistor BJT sui cui collettori si trovano i display. In questo modo riusciamo a comandare individualmente i singoli display: se vogliamo visualizzare un carattere sul display più a sinistra, dovremo inviare il codice del carattere sulla porta B e portare alto soltanto RA0. Ovviamente soltanto un display alla volta potrà essere acceso. Occorre quindi accenderli in successione ad una velocità tale da non farne percepire all'occhio lo spegnimento. Per ottenere questo effetto è sufficiente accendere ogni cifra almeno ogni 20ms. Dal momento che le cifre sono 4, dovremo tenerle accese singolarmente meno di 5ms. Per evitare un effetto di "sfarfallamento" possiamo anche usare una velocità maggiore (senza esagerare, altrimenti otterremo una bassa luminosità).
Riassumendo:
• Sulla porta B dovranno essere inviati ciclicamente i valori corrispondenti alle 4 cifre che vogliamo visualizzare.

• Sulle porta A dovrà essere portata alta una sola linea alla volta, quella corrispondente alla cifra da visualizzare (avremo quindi ciclicamente 0b0001, 0b0010, 0b0100, 0b1000). Il codice che implementa questo comportamento è il seguente:

```
void main() {
char n, ds;
char segs[16]={0x3F, 0x06,
0x5B, 0x4F, 0x66, 0x6D,
0x7D, 0x07, 0x7F, 0x6F,
0x77, 0x7C, 0x39, 0x5E,
0x79, 0x71};
char cifra[4]={1,2,3,4};
// Inizializz. porte
TRISA = 0;
TRISB = 0;
// Loop infinito
while(1) {
ds = 1;
for(n=0; n<4; n++) {
PORTB = segs[cifra[n]];
PORTA = ds;
Delay_ms(2);
ds = ds << 1;
}
}
}
```

Il codice somiglia molto a quello visto in precedenza, le uniche differenze riguardano l'utilizzo di un altro array (*cifra[…]*), che contiene le cifre da visualizzare nei 4 display, e l'impiego della porta A per selezionare il display. Il loop principale contiene un ciclo **for**, che viene ripetuto per 4 volte (una per ogni cifra), al cui interno viene impostato il valore della cifra corrente sulla porta B, ed aggiornato il contenuto della porta A. In pratica all'inizio la porta A conterrà il valore 0b0001, alla porta B viene assegnato il valore adatto a rappresentare la prima cifra, si attende 2ms, al valore della porta A viene applicato uno scorrimento a sinistra di un bit, ottenendo 0b0010, ed il ciclo *for* riprende con la seconda iterazione, in cui viene assegnato alla porta B il codice per la seconda cifra. Si prosegue così fino alla quarta, e quindi si esce dal ciclo *for*, ma non dal loop principale, che si ripeterà all'infinito. Sul display dovrebbero apparire le cifre 1234 indicate nell'array *cifra*. A questo punto potremo pensare di sfruttare il display realizzato per implementare delle funzioni utili. Se ad esempio riuscissimo a visualizzare il valore di una qualsiasi variabile, potremmo usare il display come un comodo dispositivo di output (sia nelle applicazioni, sia in fase di debug). Per fare questo possiamo utilizzare alcune funzioni messe a disposizione dalle librerie del MikroC, ed in particolare quelle che permettono di creare delle stringhe a partire dal valore numerico di una variabile. Una di queste funzioni è la seguente:

```
void ShortToStr(short number,
char *output);
```

La funzione prende in ingresso una variabile di tipo *short*, ed un puntatore ad un array di 4 caratteri (come il nostro *cifra[…]*), e restituisce nell'array di caratteri una stringa

corrispondente al numero indicato come primo parametro. Ad esempio:

```
short t = -24;
char txt[4];
ShortToStr(t, txt);
```

In questo caso l'array *txt[…]* conterrà i seguenti 4 caratteri " –24" (il primo carattere è uno spazio). Esistono anche le funzioni IntToStr, WordToStr, FloatToStr, che lavorano nello stesso modo, ma usano tipi di variabili diversi e restituiscono un numero maggiore di caratteri. Possiamo utilizzare questa funzione con il codice fino ad ora scritto? Si, ma dobbiamo apportare qualche piccola modifica. La funzione infatti restituisce una stringa, i cui valori sono codificati in ASCII. Se diamo un'occhiata alla tabella dei codici ASCII (menu **Tools/ASCII Chart**) vediamo che i codici che ci vengono restituiti vanno dal 48 al 57 (decimale) per i caratteri numerici, ma comprendono anche il 32 dello spazio, ed il 45 del segno "–". Se escludiamo i due segni non numerici, i codici dei caratteri numerici possono essere ottenuti semplicemente sottraendo 48 al codice ASCII (o anche considerando solo i 4 bit meno significativi, cioè eseguendo un AND con 0x0F). Il codice diventa quindi il seguente:

```
void main() {
char n, ds, val;
char segs[16]={0x3F, 0x06,
0x5B, 0x4F, 0x66, 0x6D,
0x7D, 0x07, 0x7F, 0x6F,
0x77, 0x7C, 0x39, 0x5E,
0x79, 0x71};
char cifra[4];
val=-47;
// Inizializz. porte
TRISA = 0;
TRISB = 0;
// Conversione
ShortToStr(val, cifra);
// Loop infinito
while(1) {
ds = 1;
for(n=0; n<4; n++) {
PORTB=segs[cifra[n]-48];
if (cifra[n]==32)PORTB=0x00;
if (cifra[n]==45)PORTB=0x40;
PORTA = ds;
Delay_ms(2);
ds = ds << 1;
}
}
}
```

In pratica è stata aggiunta soltanto la variabile *var*, la chiamata alla funzione ShortToStr, ed alcuni controlli nel loop che visualizza i dati. In particolare all'inizio è visualizzata la cifra (ottenuta dal codice ASCII meno 48), poi viene verificato se il carattere è uno spazio o un segno meno, ed in questi casi viene corretto l'output sulla porta B con i caratteri corrispondenti (tutti i segmenti spenti nel caso dello spazio, o solo quello centrale acceso

nel caso del meno). Poi viene assegnato il valore appropriato alla porta A, come accadeva nell'esempio precedente. A questo punto possiamo visualizzare il contenuto di qualsiasi variabile del nostro programma. Un'ultima nota interessante: non abbiamo mai utilizzato il punto decimale presente nei display. Se lo volessimo accendere in una qualsiasi delle quattro cifre, sarebbe sufficiente attivare il bit più significativo della porta B. Questo può essere ottenuto eseguendo un OR (operatore "|" del C) tra il codice del carattere attualmente visualizzato ed il valore binario 0b10000000 (cioè 0x80 esadecimale):

```
if (punto) PORTB=PORTB|0x80;
```

TIMER ED INTERRUZIONI

Il precedente esempio ha mostrato come sia possibile utilizzare un display a 7 segmenti per visualizzare il contenuto di una variabile. Come abbiamo visto per ottenere una buona visualizzazione delle 4 cifre è necessario rispettare delle precise temporizzazioni. Risulta quindi piuttosto difficile aggiungere altre funzioni al programma visto prima, perché qualsiasi altro compito inserito nel loop principale altererebbe le temporizzazioni del display, e risulterebbe a sua volta rallentato dai tempi di attesa utilizzati per il display. Per risolvere questo problema è necessario separare e rendere indipendenti le temporizzazioni e le funzioni di gestione del display dalla normale esecuzione del programma: questo può essere ottenuto utilizzando il timer del PIC e le interruzioni. Il meccanismo è molto semplice: il PIC ha un timer ad 8 bit (chiamato Timer0 o TMR0) che conta da 0 ad 0xFF (in totale 256 valori), ed è incrementato ad ogni ciclo macchina (4 cicli di clock) o suoi multipli. Quando il valore del timer supera 0xFF, può essere generata un'interruzione. Utilizzando il timer è quindi possibile richiamare un'apposita funzione (la routine d'interruzione) a intervalli di tempo prestabiliti. Nel nostro caso il codice all'interno della routine dovrà aggiornare il display un digit alla volta. Occorrerà quindi impostare il timer ed il suo prescaler in modo da generare un interrupt ogni 2ms. Per abilitare le interruzioni associate al TMR0 dobbiamo impostare ad 1 i bit GIE e T0IE del registro INTCON, ed abilitare il prescaler col fattore di divisione voluto sul registro OPTION_REG. Se abbiamo una frequenza di clock di 4MHz, possiamo ottenere un'interruzione ogni 2ms usando un fattore di divisione pari a 8 (infatti 4*256*8/4MHz=2ms). Dopo avere impostato il corretto valore nei registri, dobbiamo scrivere la routine di gestione delle interruzioni. Per fare questo sarà sufficiente creare una funzione chiamata "interrupt", il MikroC la riconoscerà come routine di gestione delle interruzioni, e la richiamerà automaticamente in caso di interruzioni. Il codice risultante è il seguente:

```
char n, ds, cifra[4];
char segs[16]={0x3F, 0x06,
0x5B, 0x4F, 0x66, 0x6D,
0x7D, 0x07, 0x7F, 0x6F,
0x77, 0x7C, 0x39, 0x5E,
0x79, 0x71};
void main() {
short val;
// Inizializz. porte
TRISA = 0;
TRISB = 0;
n=0;
ds=1;
val=-125;
ShortToStr(val, cifra);
OPTION_REG = 0x82;
```

```
TMR0 = 0;
INTCON = 0xA0;
while(1) {
}
}
void interrupt()
{
PORTA=ds;
PORTB=segs[cifra[n]-48];
if (cifra[n]==32) PORTB=0x00;
if (cifra[n]==45) PORTB=0x40;
n++;
ds <<= 1;
if (n > 3) {
n = 0;
ds = 1;
}
// Reset del timer
TMR0 = 0;
// Reset flag TMR0IF/TMR0IE
INTCON = 0x20;
}
```

Le differenze rispetto alla versione precedente sono essenzialmente 3. Innanzi tutto alcune delle variabili utilizzate prima sono state dichiarate fuori dalla funzione *main*, quindi risultano essere delle variabili globali: tutte le funzioni possono leggerle e modificarle. Questo risulta utile dal momento che devono essere inizializzate o modificate all'interno della funzione *main*, ma devono essere lette ed utilizzate dentro la routine d'interruzione, inoltre il loro valore deve essere mantenuto anche quando si esce dalle rispettive funzioni. In questo modo all'interno del loop principale si potrà assegnare un valore all'array *cifra*, e la routine d'interruzione provvederà a visualizzarlo indipendentemente. La funzione *main* è quasi vuota: contiene solamente le istruzioni di inizializzazione dei registri e delle variabili, e finisce con un loop infinito vuoto, che in pratica attende solamente il verificarsi delle interruzioni. La routine d'interruzione invece svolge le stesse funzioni viste prima, solo che opera su una cifra alla volta: aggiorna il valore di PORTA e PORTB come nel programma precedente, modifica il valore di *n* e *ds* in modo da puntare al prossimo carattere, e prima di terminare resetta il valore del timer e del suo flag d'interruzione. Non è necessario utilizzare la funzione di ritardo dal momento che i valori impostati sulle porte saranno mantenuti fino alla prossima interruzione, che si verificherà dopo circa 2ms. A questo punto sarà possibile eseguire altre operazioni nel loop principale e modificare come si vuole l'array *cifra[]*, ed il display sarà aggiornato di conseguenza ed in maniera automatica. Ad esempio si può realizzare un conteggio inserendo nel loop il seguente codice:

```
while(1) {
ShortToStr(val, cifra);
Delay_ms(250);
val++;
}
```

Notare che in questo caso il ritardo di 250ms inserito non influisce completamente sull'aggiornamento del display, che continua a funzionare con le sue temporizzazioni!

Illuminotronica 2014 si fa in tre

L'evento professionale italiano dedicato a lighting, elettronica e building automation

Tante le novità di **Illuminotronica 2014**, l'unica mostra-convegno italiana sul mondo delle nuove tecnologie per l'illuminazione a LED e la domotica in programma a **Padova** dal **9** all'**11 ottobre**. Un appuntamento che si preannuncia ricco di contenuti, corsi, convegni e iniziative speciali dedicati a un pubblico specializzato di professionisti. Un momento, inoltre, per formarsi e informarsi, per aggiornarsi e confrontarsi, per incontrare e contattare gli esperti del mercato.

Le aree espositive di Illuminotronica

Tra le novità di quest'anno, la divisione della mostra-convegno in tre grandi aree espositive, per offrire una visione completa sui settori dell'elettronica, dell'illuminazione e della building automation, tre ambiti sempre più interconnessi e integrati tra loro.

1. Illuminazione a LED ed elettronica per il lighting
È l'area storica e più importante di Illuminotronica che vede la partecipazione sia di produttori di corpi illuminanti *(indoor e outdoor)* sia di aziende che producono e forniscono componenti e sottosistemi per il lighting (LED/OLED, ottiche, driver, connettori, alimentatori, dissipatori ma anche sensori, tecnologie wireless ecc.).

2. Domotica e Building Automation
Un'area nuova, molto vicina al mondo della luce, dove vedere e toccare con mano le soluzioni più innovative per il controllo e la gestione *"intelligente"* dell'edificio.

3. Elettronica Micro
Il rinnovato punto di incontro per chi opera nell'elettronica industriale, per chi progetta e produce in Italia componenti, attrezzature e sistemi. Un'area promossa da Assodel per valorizzare la pervasività dell'elettronica nel rapporto con la crescente esigenza di efficienza energetica.

Iniziative speciali

Tra le iniziative speciali, un **Laboratorio per gli installatori**, con prove tecniche e

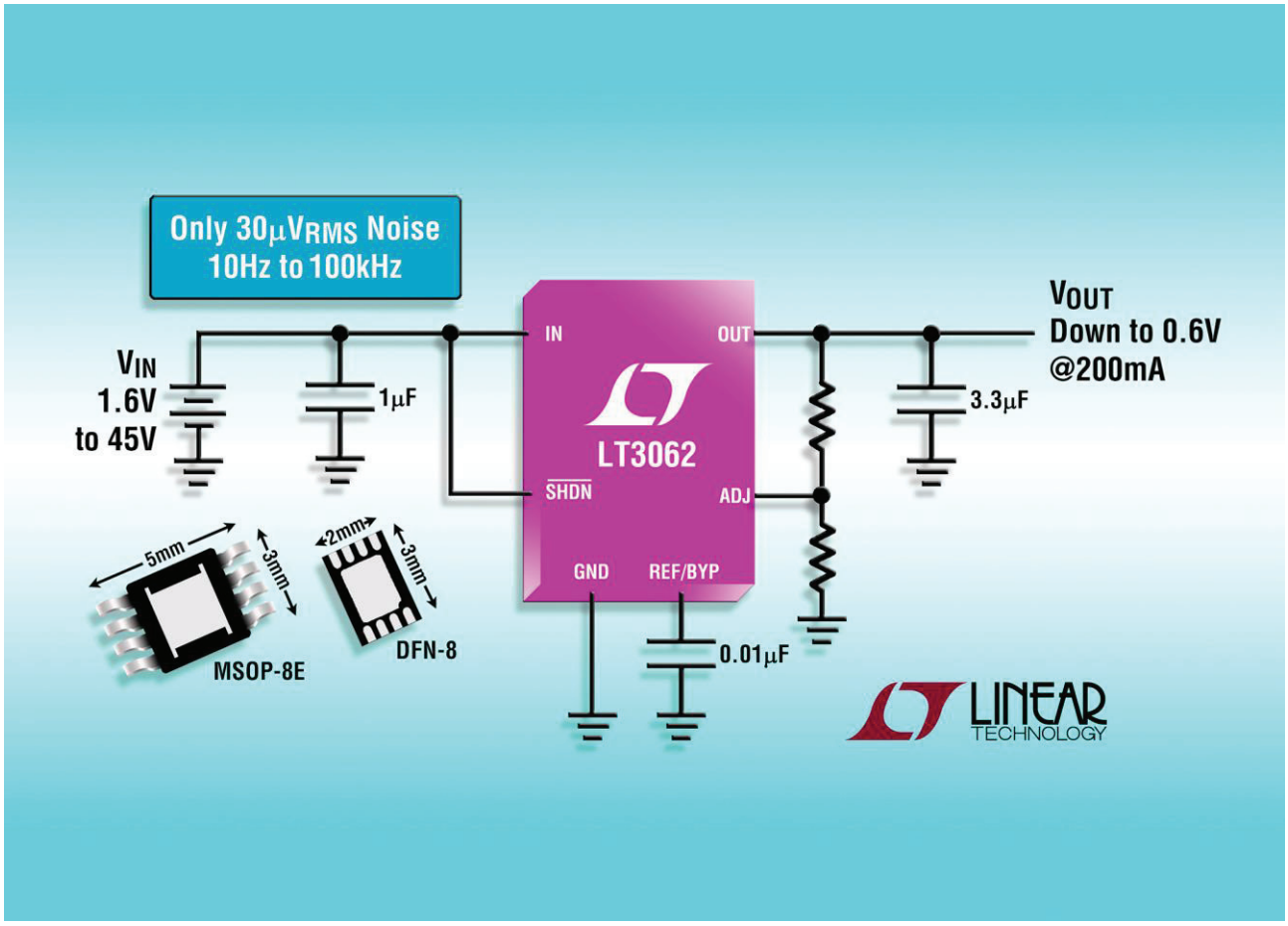
dimostrazioni sull'utilizzo e sul controllo dei LED, sulle tecnologie per il lighting e la domotica. Un'**area demo** dove ricreare gli effetti benefici della luce a LED e della domotica in spazi da interno (gli ambienti di una casa/albergo) e in una strada e dove toccare con mano le soluzioni più innovative. Un angolo (**Codega Corner**), inoltre, dove incontrare gli esperti del settore e confrontarsi con le loro esperienze e idee.

Ma anche premi – come il **Premio Codega** alle eccellenze nel lighting designer e il **LEDin Award** ai giovani designer e alle idee più innovative nella progettazione di corpi illuminanti – corsi di formazione di vario livello tecnico, talk show, convegni e momenti conviviali.

Per maggiori informazioni:
www.illuminotronica.it

LDO da 45 V

con un rumore di 30µVRMS



Linear Technology Corporation presenta l'LT3062, un regolatore di tensione lineare ad alta tensione, basso rumore e basso dropout. Il circuito integrato fornisce fino a 200mA di corrente di uscita continua con una tensione di dropout di 300mV a pieno carico. L'LT3062

presenta un ampio range di tensioni di ingresso compreso tra 1,6V e 45V, per fornire tensioni di uscita regolabili comprese tra 0,6V e 40V. Un unico condensatore garantisce un funzionamento programmabile a bassissimo rumore – solo 30µVRMS su una larghezza di banda di 10Hz-100kHz – e funzionalità di "soft start" del riferimento, al fine di impedire l'overshoot della tensione di uscita all'accensione. Il valore di tolleranza della tensione di uscita dell'LT3062 è altamente preciso e corrisponde al ±2% rispetto a linea, carico e temperatura. L'ampio range di tensioni di ingresso e di uscita del dispositivo, la rapida risposta al transitorio, la bassa corrente di riposo di soli 40µA (in modalità operativa) e <1µA (in modalità di arresto) rendono l'LT3062 il dispositivo ideale per sistemi portatili alimentati a batteria che richiedono una durata ottimale e per l'uso nelle applicazioni di alimentazione dei settori automotive, industriale e avionico. Il dispositivo funziona con condensatori ceramici di uscita di dimensioni ridotte e a basso costo che ottimizzano la stabilità e la risposta al transiente. È stabile con un condensatore di uscita minimo da 3,3µF. I condensatori esterni di dimensioni ridotte possono essere utilizzati senza ulteriori resistenze in serie (ESR), requisito spesso necessario di molti altri regolatori. Il dispositivo è anche dotato di un circuito interno che offre protezione in caso di inversione della polarità della batteria, dell'uscita e della corrente, limitazione della corrente con foldback in base a un valore predefinito e limitazione della temperatura. L'LT3062 viene offerto in package DFN e MSOP di 2 x 3mm e a 8 conduttori per un ingombro ridotto. Le versioni di grado E e I hanno una temperatura di giunzione tra –40°C e +125°C, la versione di grado H (elevata disponibilità) è specificata per il range di temperature compreso tra –40°C e +150°C, mentre la versione MP (militare) funziona nel range da –55°C a +150°C. Il prezzo per 1.000 pezzi parte da $1,70/cad. per la versione E nei package DFN e MSOP. Tutte le versioni sono disponibili a magazzino.
Per maggiori informazioni, visitare la pagina www.linear.com/product/LT3062

Toshiba presenta lo starter kit TZ5000 App Lite

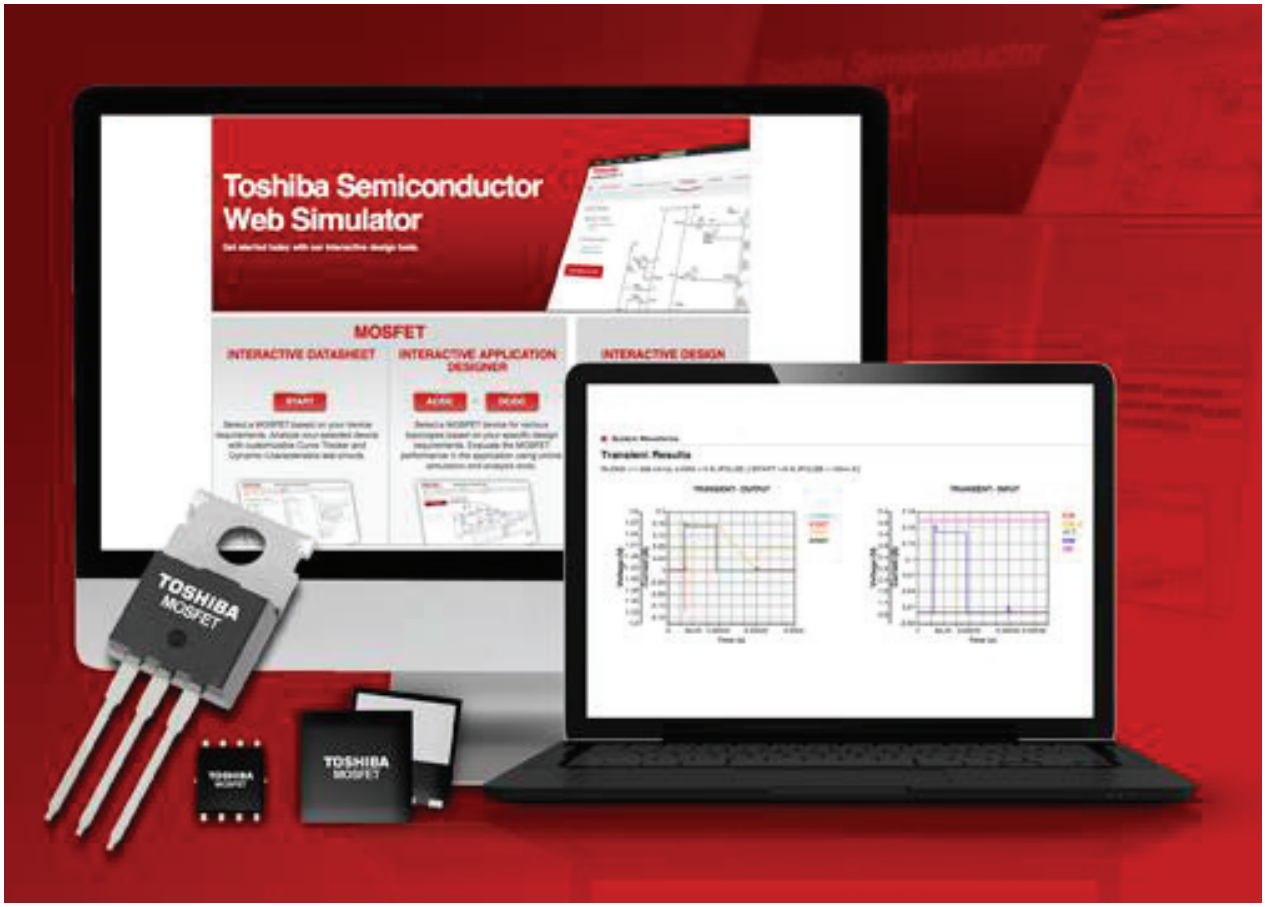
Toshiba Electronics Europe annuncia un tool online che aiuta i progettisti ad analizzare le prestazioni dei componenti utilizzati negli stadi di potenza dei circuiti embedded e a migliorare l'efficacia e l'efficienza generali di un sistema elettronico. Il Toshiba Semiconductor Web Simulator permette agli utenti di simulare le prestazioni dei MOSFET in diverse condizioni di tensioni e temperatura, nonché di analizzare le forme d'onda relative alla commutazione in convertitori AC/DC e DC/DC di diverse topologie, tra cui full-bridge, flyback e conversione buck sincrona. Oltre ai MOSFET, è possibile analizzare il comportamento di regolatori a bassa caduta di tensione (LDO, *Low Drop-Out*) e gli interruttori di carico integrati. Essendo basati su cloud, i dati e i progetti possono essere condivisi facilmente tra i team di progettazione, a prescindere da dove essi si trovino. Inoltre, il simulatore può essere scaricato e utilizzato in modalità off-line, permettendo una progettazione circuitale ancora più versatile.

Il tool si compone di tre elementi interattivi: *datasheet*, *application designer* e *design note*. I primi due elementi riguardano i MOSFET, gli altri i regolatori LDO e gli interruttori di carico. Con il datasheet di MOSFET interattivo, il progettista può scegliere i componenti da una lista e analizzare la scelta effettuata utilizzando un tracciatore di curve personalizzabile e circuiti di prova per le caratteristiche dinamiche. Per semplificare la scelta, la ricerca può essere raffinata regolando parametri come tipo di contenitore, resistenza e gamma di tensioni.

Utilizzando il designer applicativo interattivo è possibile scegliere un dispositivo MOSFET da utilizzare in diverse configurazioni topologiche, sulla base di specifici requisiti di progetto. È quindi possibile valutare le prestazioni del componente nel sistema utilizzando i tool di analisi e di simulazione disponibili online. Al termine della simulazione e dei confronti, viene generato un rapporto riepilogativo che comprende uno schema circuitale e una lista di materiali. Questo documento può essere stampato, salvato e condiviso con altri colleghi. È anche possibile esportare i modelli di progetto dei MOSFET in altri ambienti di simulazione.

La nota di progetto interattiva offre funzioni simili per i regolatori LDO e i commutatori sotto carico integrati. Gli utenti possono identificare i componenti in base ai parametri scelti, creare un progetto circuitale con un solo clic, analizzarlo e generare una lista di materiali con un rapporto riepilogativo.

Per registrarsi e iniziare a utilizzare il Toshiba Semiconductor Web Simulator, fare clic qui oppure collegarsi alla pagina
http://www.semicon.toshiba.co.jp/eng/design_support/simulation/index.html



Nuovo nato in casa Microchip

Microchip annuncia un nuovo ingresso nella sua fa Famiglia di microcontroller PIC12/16LF155X 8-bit (MCU), i dispositivi PIC16LF1554 e PIC16LF1559 (PIC16LF1554/9). Il PIC16LF1554/9 contiene due *Analogue-to-Digital Converter* (ADC) indipendenti con campionature da 100K al secondo 10-bit, con supporto per *Capacitive Voltage Divider* (CVD) hardware per rilevamento tattile capacitivo. Questa eccezionale configurazione ADC permette una maggiore efficienza di acquisizione da parte di sensori ed è grandemente di sostegno con tecniche avanzate di *touch-sensing* in ambienti fortemente rumorosi, applicazioni *low-power*, tastiere a matrice e progetti di applicazioni resistenti all'acqua.

L'MCU PIC16LF1554/9 14- e 20-pin combina fino a 17 canali ADC con moduli CVD automatici per implementare rilevamento capacitivo e altre applicazioni di campionamento front-end con sovraccarico software minimale. In particolare, il CVD hardware riduce il codice per implementazioni di rilevamento tattile capacitivo di oltre il 40%. I dispositivi includono anche Flash fino a 14 KB / RAM 512 Byte, un oscillatore interno a 32 MHz e due moduli PWM, oltre a I^2C™, SPI e EUSART per le comunicazioni. Inoltre sono conformi a *eXtreme Low Power* (XLP) con correnti *active* e *sleep* rispettivamente di 35 µA/MHz e 30 nA, per applicazioni dove la conservazione dell'energia è essenziale. Queste caratteristiche combinate con il basso prezzo ed il piccolo footprint dei PIC16LF1554/9, li

rende adatti per una vasta gamma di applicazioni nell'elettronica di consumo, come telecomandi, riproduttori audio, e accessori per cellulari, piccoli elettrodomestici, dispositivi indossabili tra cui cuffie, orologi, e bracciali da fitness; in applicazioni medicali come monitor di pressione sanguigna e battito cardiaco indossabili; nel settore automotive per controlli e rilevamenti di bordo e pannelli di controllo; e nel settore industriale in applicazioni di RFID e sensori; oltre a molte altre applicazioni.
La completa suite Microchip di strumenti di sviluppo supporta gli MCU PIC16LF1554/9, ivi compreso anche l'*MPLAB® X Integrated Development Environment* (IDE); PICkit™ 3 (PG164130) offerto al un prezzo di 44,95$; *MPLAB XC8 Compiler* per MCU PIC 8-bit e *MPLAB® Code Configurator*.

Gli MCU PIC16LF1554 sono disponibili da subito sia in campionature che in quantità per volumi di produzione nei package 14-pin PDIP, TSSOP, SOIC e 16-pin QFN (4 x 4 x .9 mm). Gli MCU PIC16LF1559 sono disponibili da subito sia in campionature che in quantità per volumi di produzione nei package 20-pin PDIP, SSOP e QFN (4 x 4 x .9 mm).

Maggiori informazioni su: http://www.microchip.com/get/08KC

Proteggere gli Investimenti in R&D con l'Autenticazione Sicura

di Scott Jones - Executive Director, Embedded Security, Maxim Integrated

Le soluzioni di autenticazione sicura offerte da Maxim consentono agli sviluppatori di proteggere i loro sistemi dagli inevitabili tentativi di contraffazione che prendono di mira accessori e sottosistemi. La disponibilità di una memoria utente a prova di manomissione, inoltre, fornisce metodi sicuri per abilitare o disabilitare le funzioni dei sistemi con funzionalità configurabile. In questo articolo vedremo come un piccolo ma potente pezzetto di silicio possa fare una grande differenza nel bilancio economico d'impresa.

Introduzione

Nell'era dei furti di identità e degli identificativi fasulli, è estremamente importante poter contare su identificazioni certe. Questo vale non solo per le persone, ma anche per i prodotti elettronici di ogni tipo o quasi. I fabbricanti, infatti, hanno la necessità di proteggere i loro prodotti dai componenti contraffatti che le aziende produttrici di accessori e ricambi (aftermarket) tentano di introdurre nella loro catena di fornitura OEM.

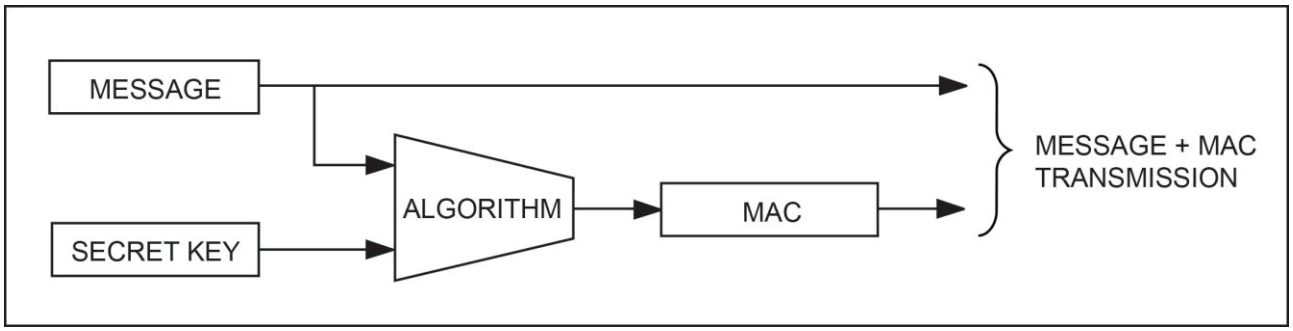


Figura 1. Modello di calcolo del MAC.

L'autenticazione sicura costituisce un'efficace soluzione di tipo elettronico per affrontare questa minaccia, e permette inoltre di aggiungere funzioni utili al prodotto finito. Questo articolo illustra il concetto dell'autenticazione e in particolare la soluzione messa a punto da Maxim – con i componenti definiti "autenticatori sicuri" - per soddisfare una gamma di requisiti applicativi comprendente la protezione della proprietà intellettuale, la gestione delle licenze HW/SW embedded, l'impostazione sicura delle soft-feature e dello stato, la conservazione dei dati a prova di manomissione.

Cos'è l'autenticazione?

L'autenticazione è un processo finalizzato a fornire la prova dell'identità nella relazione tra due o più entità. Nel caso di autenticazione unidirezionale, una sola entità fornisce prova della propria identità ad un'altra. Nell'autenticazione bidirezionale, la prova dell'identità viene fornita reciprocamente da ciascuna entità all'altra. Il metodo di autenticazione più comunemente utilizzato è la password. La principale limitazione delle password è che al momento dell'uso esse vengono esposte all'osservazione, il ché le rende vulnerabili allo spionaggio.

Dopo aver passato in rassegna la storia della crittografia, nel 1883 il linguista fiammingo Auguste Kerckhoffs pubblicò le proprie teorie in un articolo pionieristico sulla crittografia militare. Kerckhoffs sosteneva che anziché fare affidamento sulla segretezza (oscurità) del sistema, la sicurezza dovesse basarsi sulla protezione delle chiavi, poiché in caso di violazione sarebbe stato sufficiente sostituire solo queste ultime, non l'intero sistema.

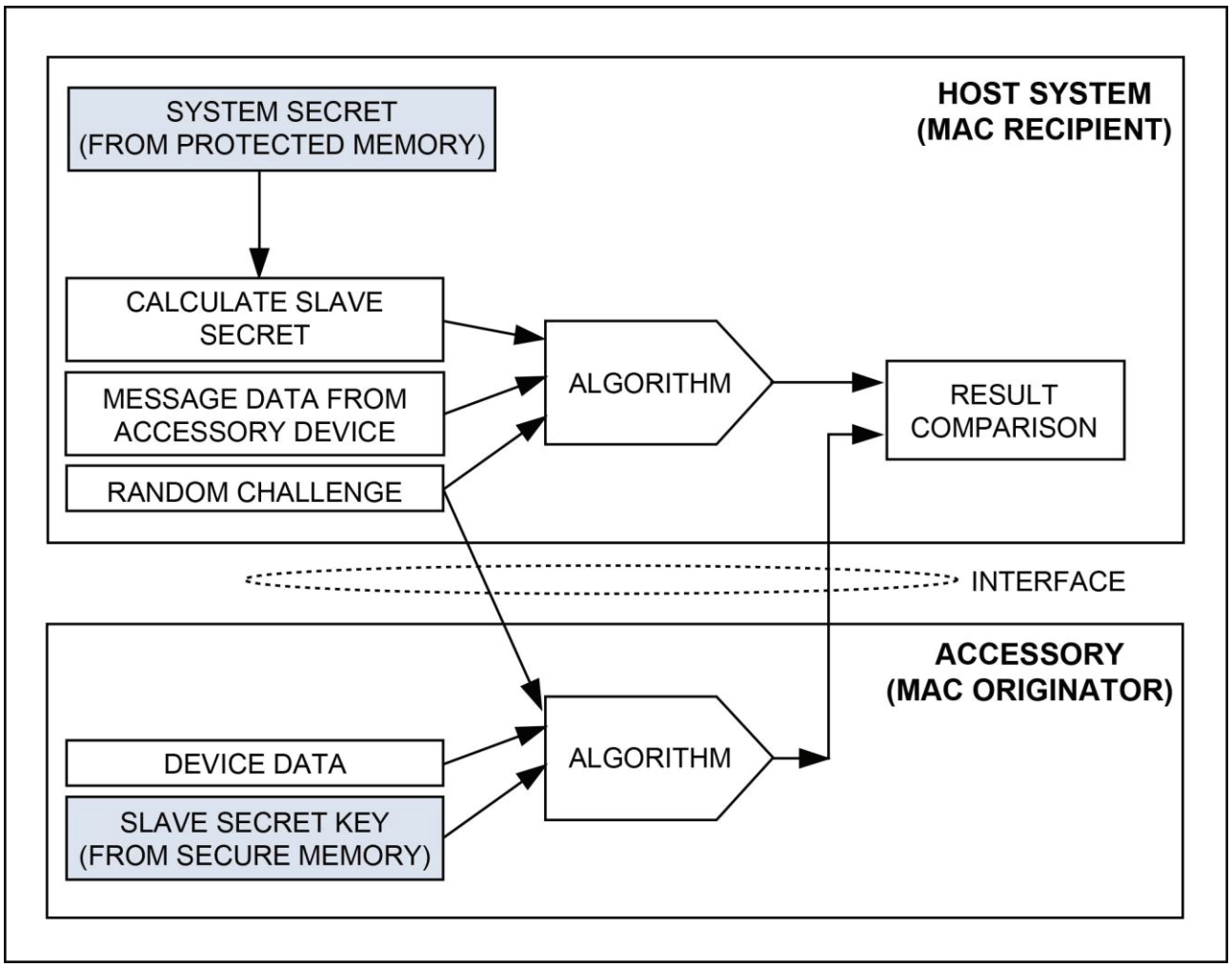


Figura 2. Flusso dei dati nell'autenticazione sfida-risposta.

Un efficace metodo di autenticazione simmetrica basato su chiavi funziona come illustrato in figura 1: la chiave segreta e i dati da autenticare ("messaggio") vengono utilizzati come input per calcolare un codice di autenticazione del messaggio (Message-Authentication-Code, MAC). Il MAC viene quindi unito al messaggio e trasmesso su richiesta. Il ricevente effettua lo stesso calcolo e confronta la propria versione del MAC con quella che ha ricevuto insieme al messaggio. Se i due MAC coincidono, il messaggio è autentico. Il punto debole di questo modello di base, tuttavia, è che un messaggio ed un MAC statici, intercettati da un aggressore, possono essere successivamente replicati da un mittente non autentico ed essere considerati autentici.
Per provare l'autenticità dell'entità che ha dato origine al MAC (ad esempio l'accessorio di un sistema), il ricevente (cioè il sistema host a cui l'accessorio è connesso) genera un numero casuale e lo invia come sfida all'originatore. L'originatore del MAC deve quindi calcolare un nuovo MAC a partire da tre elementi (la chiave segreta, il messaggio, il numero di sfida) e rispedirlo al ricevente. Se l'originatore si dimostra capace di generare un MAC valido per ogni sfida, è certamente a conoscenza della chiave segreta e quindi può essere considerato autentico. La figura 2 illustra questo flusso di autenticazione sfida-risposta e gli elementi di dati ad esso associati.
In crittografia, un algoritmo che genera un MAC di lunghezza fissa a partire da un messaggio è detto "funzione di hash unidirezionale". Il termine "unidirezionale" indica che matematicamente non è fattibile ricavare dal MAC di lunghezza fissa il messaggio iniziale, generalmente più lungo, comprendente la chiave segreta.
Due funzioni di hash unidirezionale che sono state oggetto di esame approfondito ed hanno ottenuto certificazioni internazionali sono gli algoritmi SHA-1 e SHA-2 sviluppati dal National Institute of Standards and Technology (NIST), descritti nel documento FIPS 180. Le elaborazioni matematiche su cui si basano queste funzioni sono disponibili pubblicamente nel sito web di NIST. Le caratteristiche distintive dei due algoritmi sono:
1) irreversibilità - matematicamente non è fattibile determinare l'informazione d'ingresso corrispondente a un dato MAC;
2) resistenza alle collisioni - è praticamente impossibile trovare più di un messaggio di ingresso che produca

il medesimo MAC;
3) elevato effetto valanga - ogni minimo cambiamento dell'informazione in ingresso produce un cambiamento significativo del MAC risultante. Per queste ragioni, oltre che per la serietà degli esami a cui sono stati sottoposti internazionalmente, Maxim ha scelto gli algoritmi SHA-1 e SHA-2 per l'autenticazione sfida-risposta dei propri autenticatori sicuri. Nei propri prodotti più recenti, in particolare, l'azienda ha implementato una variante di SHA-2 denominata SHA-256.

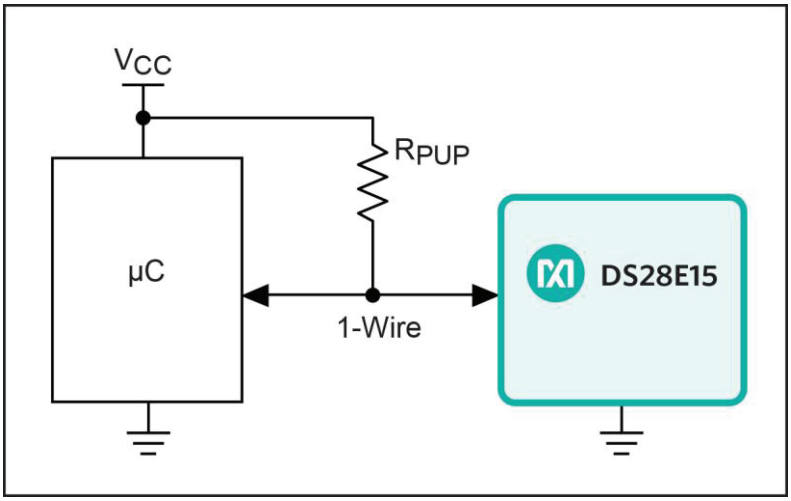


Figura 3. Esempio di applicazione base.

Autenticazione sicura a basso costo: l'implementazione del sistema
Grazie all'interfaccia 1-Wire®, qualunque sistema dotato di capacità di elaborazione digitale – es. di un microcontrollore (µC) - può essere facilmente equipaggiato con un autenticatore sicuro, come il DeepCover® Secure Authenticator (DS28E15). Nel caso più semplice, è sufficiente disporre di un pin di porta libero nel microcontrollore ed aggiungere un resistore di pull-up per la linea 1-Wire, come illustrato in figura 3. Questo approccio può però essere potenzialmente rischioso se si impiega un microcontrollore non sicuro, che può essere studiato da un aggressore per comprendere e compromettere le funzioni di sicurezza.
In alternativa, come illustrato in figura 4, il DS28E15 può essere azionato e controllato tramite un apposito IC, come il coprocessore SHA-256 DeepCover Secure Authenticator (DS2465) con interfaccia master 1-Wire integrata. Sebbene il DS28E15 possa essere gestito anche con un approccio basato sul solo microcontrollore, l'impiego del DS2465 offre vari vantaggi:
1) solleva il µC host dal compito dei calcoli SHA-256;
2) permette di conservare in modo molto sicuro la chiave segreta SHA-256 del sistema;
3) solleva il µC host dal compito di generare la forma d'onda 1-Wire.

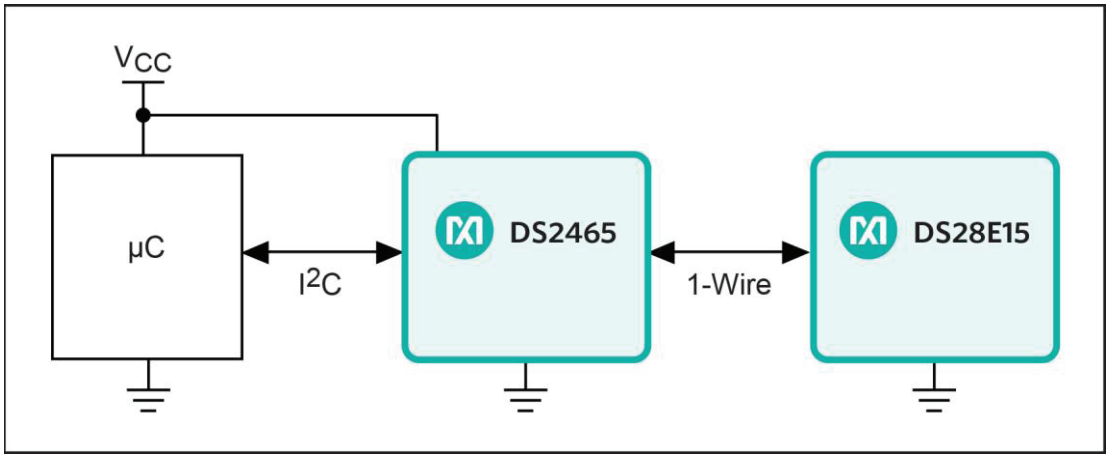


Figura 4. Uso di un coprocessore per aumentare la sicurezza.

Prevenzione delle contraffazioni
I sistemi dotati di elementi sostituibili - come sensori, periferiche, moduli o materiali di consumo - sono

comunemente presi di mira da aziende 'aftermarket' non autorizzate. Le versioni contraffatte degli elementi sostituibili possono causare preoccupazioni per la sicurezza (safety) del sistema, ridurre la qualità dell'applicazione e, in generale, avere un impatto negativo sulla soluzione OEM. Aggiungere alla soluzione una funzione di autenticazione sicura consente al sistema host di testare l'autenticità del sensore o del modulo e, se viene rilevata una contraffazione, di compiere azioni appropriate alla specifica applicazione. Come illustrato in figura 5, per confermare l'autenticità viene svolta una sequenza sfida-risposta tra il sistema e la periferica ad esso collegata.

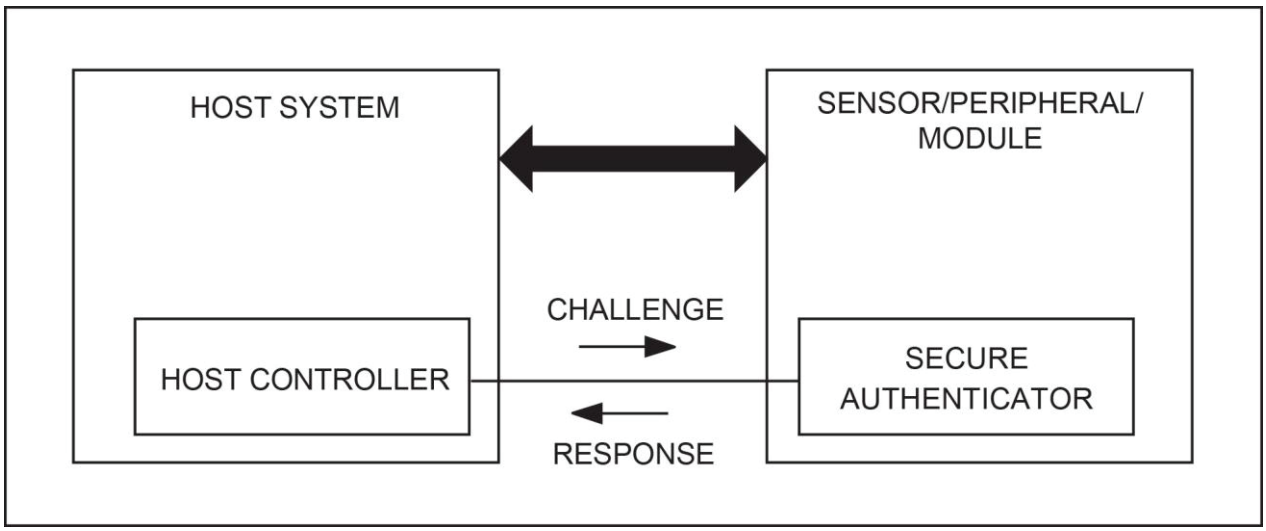


Figura 5. Test di autenticità con una sequenza sfida-risposta.

Gestione delle licenze HW/SW embedded

I "progetti di riferimento", che vengono concessi in licenza ed eventualmente si traducono in prodotti fabbricati da terze parti, richiedono barriere protettive per impedire l'uso non autorizzato della relativa proprietà intellettuale. Per il calcolo dei compensi occorre inoltre tenere traccia ed accertare il numero degli esemplari prodotti. Un autenticatore SHA-256 pre-programmato (con una chiave segreta, una memoria utente ed impostazioni installate prima della consegna al fabbricante di terza parte), come il DeepCover Secure Authenticator (DS28E25), può facilmente soddisfare questi ed altri requisiti. Il progetto di riferimento effettua un'autoverifica all'accensione (figura 6) eseguendo una sequenza di autenticazione con il DS28E25. Soltanto un DS28E25 con una chiave segreta valida, nota solo all'azienda licenziante ed all'elettronica del progetto di riferimento, è in grado di fornire in risposta un MAC valido. Se viene rilevato un MAC non valido, il processore del progetto di riferimento può compiere azioni appropriate alla specifica applicazione. Un vantaggio aggiuntivo di questo approccio è la possibilità di concedere in licenza ed abilitare selettivamente le varie funzioni del progetto di riferimento, tramite le impostazioni conservate nella memoria sicura del DS28E25 (per un approfondimento di questo concetto si veda il paragrafo "Gestione delle soft-feature").

Esistono due modalità sicure per fornire al licenziatario o al fabbricante di terza parte il DS28E25, od un altro autenticatore sicuro, dotato di una chiave segreta valida:

1) il dispositivo può essere pre-programmato dall'azienda che concede in licenza il progetto di riferimento, oppure

2) pre-programmato da Maxim secondo i criteri stabiliti dalla società licenziante e quindi fornito al fabbricante di terza parte.

In entrambi i casi, il numero dei dispositivi inviati al licenziatario o al fabbricante è noto e può essere utilizzato per calcolare i compensi dovuti per la licenza.

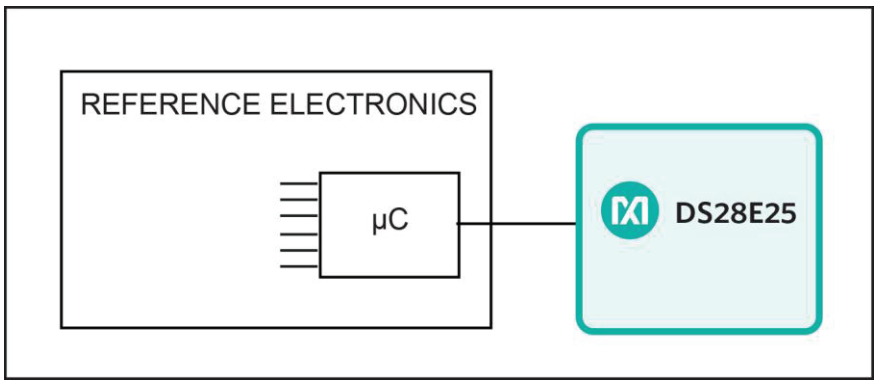


Figura 6. Autenticazione del progetto di riferimento.

Verifica dell'autenticità dell'hardware

Per quanto riguarda la verifica dell'autenticità dell'hardware occorre considerare due casi (figura 7):
1) una scheda PCB clonata contenente una copia esatta del firmware del µC o della configurazione dell'FPGA;
2) un sistema host clonato.
In questo esempio viene usato il DS28E01-100 basato su SHA-1.
Nel primo caso, il firmware o l'FPGA controlla l'autenticità della scheda PCB clonata. Affinché la verifica abbia successo, il produttore del clone deve caricare una chiave segreta in un autenticatore sicuro per scrivere i dati nella EEPROM utente. In questo modo i dati possono apparire corretti, ma tuttavia la chiave segreta non è valida nell'ambito di quel sistema. A causa delle difficoltà legate alle modifiche ed alla necessità di mantenere la compatibilità con l'host, il firmware o la configurazione dell'FPGA deve essere una copia esatta dell'originale. Se durante la fase di accensione la scheda esegue un'autenticazione sfida-risposta con il DS28E01-100, il MAC generato da questo dispositivo sarà diverso dal MAC calcolato dal firmware o dall'FPGA. Questa mancata corrispondenza dimostra chiaramente che la scheda non è autentica. Il sistema può rilevare ciò eseguendo una sequenza sfida-risposta nei confronti della scheda e può quindi compiere azioni appropriate alla specifica applicazione.
Nel secondo caso, la scheda PCB controlla l'autenticità del sistema host. La verifica può utilizzare la seguente procedura:
1) generare un numero di sfida e far calcolare al DS28E01-100 un MAC di autenticazione sfida-risposta;
2) inviare gli stessi dati utilizzati per il calcolo del MAC (tranne la chiave segreta, ovviamente) all'host della rete, che quindi calcola e restituisce un MAC di autenticazione sfida-risposta basato su quei dati e sulla propria chiave segreta.
Se i due MAC coincidono, l'host può essere ritenuto autentico dalla scheda.

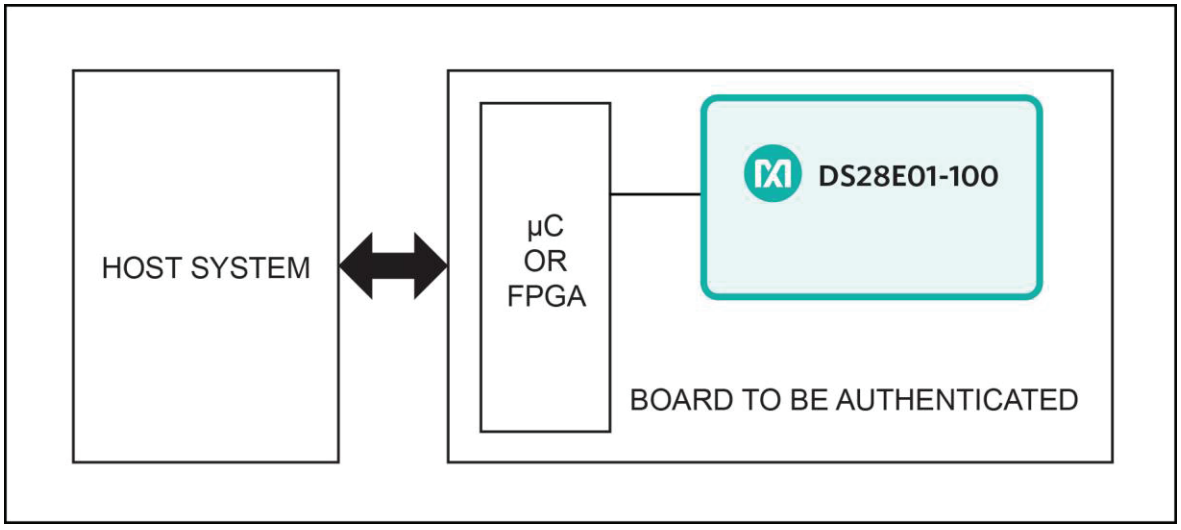


Figura 7. Esempio di autenticazione dell'hardware.

Gestione delle soft-feature

In termini di dimensioni, i sistemi elettronici coprono una gamma che va dai prodotti portatili fino ad apparati che occupano diversi rack. Maggiori sono le dimensioni, maggiore è anche il relativo costo di sviluppo. Per

tenere i costi sotto controllo, si cerca quindi di costruire i grandi sistemi partendo da una selezione limitata di sottosistemi più piccoli (schede). Spesso non tutte le funzioni di un sottosistema sono necessarie per una specifica applicazione. Anziché eliminare le funzioni superflue, è più conveniente disabilitarle tramite il software di controllo, lasciando la scheda invariata. Questa scelta, tuttavia, crea un nuovo problema: un cliente scaltro che necessiti di diversi sistemi con funzionalità completa potrebbe comprare un unico esemplare con queste caratteristiche e quindi copiarne il software su numerosi esemplari con funzionalità ridotte, di prezzo inferiore. Questi ultimi assumerebbero così tutte le caratteristiche dell'esemplare più costoso ed il fornitore del sistema riceverebbe un compenso inferiore al dovuto.
Installando su ogni scheda/sottosistema un dispositivo SHA-256 di Maxim, come il DeepCover Secure Authenticator (DS28E22), il fornitore del sistema può difendersi da questo tipo di frode. Oltre a servire per l'autenticazione sfida-risposta, lo stesso DS28E22 può conservare le impostazioni di configurazione individuali nella propria EEPROM utente. Come illustrato nel paragrafo "Sicurezza dei dati", in questo modo le impostazioni sono protette dalle modifiche non autorizzate, ed il fornitore del sistema assume il pieno controllo di questo aspetto. Le impostazioni di configurazione possono essere conservate nella forma che il progettista del sistema ritiene più appropriata, ad esempio come bitmap o come parole di codice.

Il dispositivo di autenticazione sicura

Architettura complessiva
L'engine SHA dei dispositivi SHA-1 e SHA-256 può essere azionato in tre modi diversi a seconda dell'operazione da eseguire. In tutti i casi, l'engine riceve i dati di ingresso e calcola un MAC come risultato. Per ciascun tipo di operazione ci sono specificità nei dati inviati all'engine SHA, legate all'uso previsto del MAC risultante. Il requisito fondamentale dei sistemi sicuri basati su chiave simmetrica è che, per ogni operazione SHA, l'host debba conoscere o essere in grado di calcolare la chiave segreta conservata nel dispositivo slave, al fine di essere autenticato.
Nota: date le caratteristiche di sicurezza dei prodotti di autenticazione sicura, i dettagli dei dispositivi sono stati omessi da questo documento. Altre informazioni sono presenti nelle versioni complete dei fogli specifiche dei singoli dispositivi, disponibili previo accordo di non divulgazione (Non-Disclosure-Agreement, NDA).

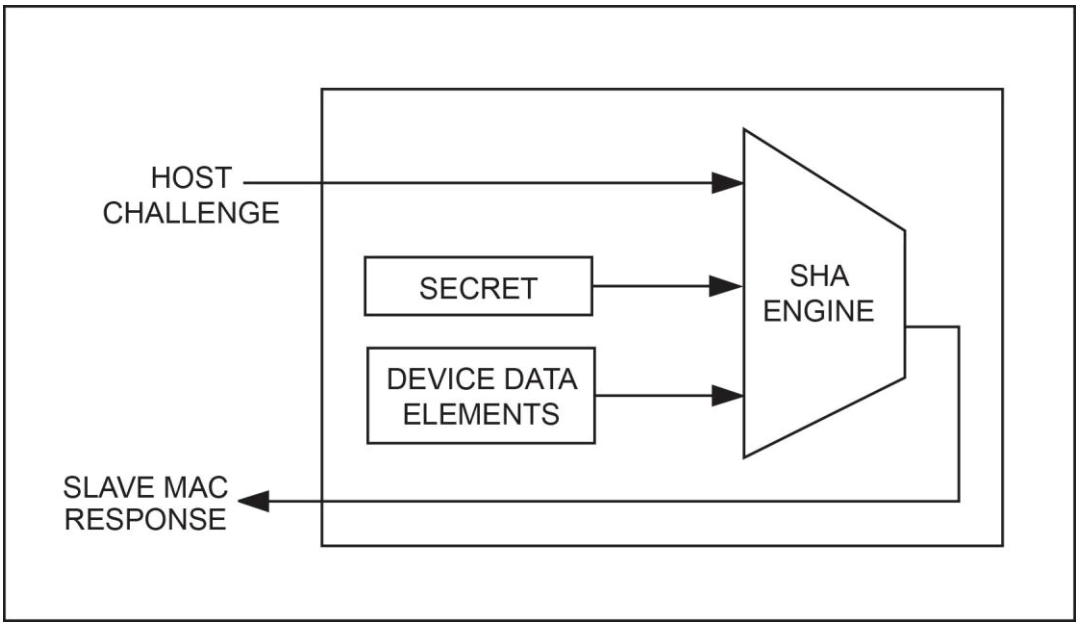


Figura 8. Flusso dei dati per il MAC di autenticazione sfida-risposta.

MAC di autenticazione sfida-risposta
La funzione principale degli autenticatori sicuri SHA-1 e SHA-256 è l'autenticazione sfida-risposta. L'host invia un numero casuale di sfida e incarica il dispositivo slave di calcolare un MAC di risposta, a partire dai diversi elementi che insieme costituiscono il "messaggio" (figura 8): il numero di sfida, la chiave segreta, la memoria utente e dati aggiuntivi.
Al termine del calcolo, il dispositivo slave invia il proprio MAC all'host per la verifica. L'host quindi ripete il calcolo del MAC usando una chiave segreta valida e gli stessi dati di messaggio che sono stati usati dallo slave. La coincidenza con il MAC ricevuto dallo slave dimostra l'autenticità del dispositivo, poiché solo uno

slave autentico può rispondere correttamente alla sequenza sfida-risposta. È cruciale che la sfida sia basata su dati casuali. Un numero di sfida che non cambia mai apre la strada ad attacchi ripetitivi (replay) che usano un MAC statico, valido, registrato e replicato, anziché un MAC calcolato all'istante da uno slave autentico.

Sicurezza dei dati

Oltre a provare l'autenticità, è estremamente desiderabile garantire che i dati conservati nel dispositivo slave siano affidabili. Per questa ragione, l'accesso di scrittura alla EEPROM dell'autenticatore sicuro è soggetto a restrizioni. Prima di copiare i dati dal buffer di ingresso alla EEPROM o ai registri di controllo, il dispositivo slave esige che l'host richiedente dia prova della propria autenticità fornendo un valido MAC di autenticazione per accesso di scrittura. Il dispositivo slave calcola questo MAC a partire dai nuovi dati nella propria memoria buffer di ingresso, dalla propria chiave segreta e da dati aggiuntivi, come illustrato in figura 9.

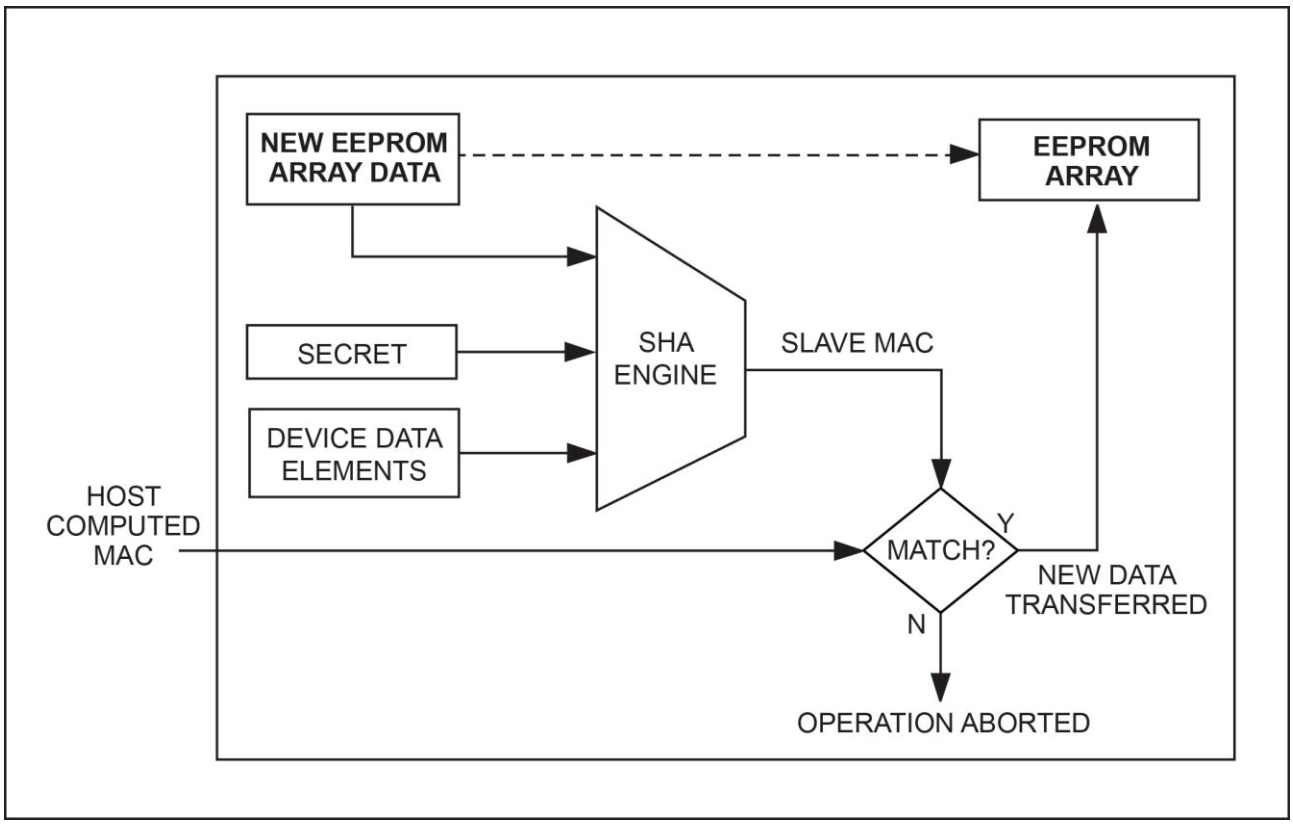


Figura 9. Flusso dei dati per un MAC di autenticazione per accesso di scrittura.

Un host autentico conosce, o è in grado di calcolare, la chiave segreta e può generare un MAC valido per l'accesso di scrittura. Quando riceve il MAC dall'host durante il comando di copia, lo slave lo confronta con il proprio risultato. I dati vengono trasferiti dal buffer di ingresso alla loro destinazione nella EEPROM solo se i due MAC coincidono. Ovviamente le pagine di memoria a scrittura protetta non possono essere modificate, nemmeno se il MAC è corretto.

Protezione della chiave segreta

L'architettura degli autenticatori sicuri di Maxim consente di caricare direttamente la chiave segreta nel dispositivo. La chiave segreta è protetta contro la lettura e, se richiesto, anche contro la scrittura; in quest'ultimo caso la chiave non può mai essere cambiata. Questo meccanismo di protezione è efficace a patto che durante l'installazione iniziale, nel sito produttivo dell'apparato, l'accesso alla chiave segreta sia sicuro e controllato.

La protezione della chiave segreta può essere aumentata in vari modi:

1) assegnando al dispositivo slave il compito di calcolare la propria chiave segreta;
2) suddividendo questo calcolo in più fasi effettuate in siti diversi;
3) creando chiavi segrete esclusive legate al singolo dispositivo, includendo nel calcolo il numero che identifica univocamente ogni esemplare;
4) una combinazione delle modalità 2 e 3.

Se ogni autenticatore sicuro calcola la propria chiave segreta, i relativi “ingredienti” sono resi noti ma la chiave vera e propria non viene mai esposta. Se la chiave segreta viene calcolata in più fasi presso siti diversi, in ciascun sito vengono resi noti solo gli ingredienti usati localmente, il che fornisce un metodo per controllare la divulgazione della chiave segreta finale. Se si creano chiavi segrete esclusive legate a ogni singolo esemplare del dispositivo, l’host deve compiere un calcolo aggiuntivo ma diviene possibile minimizzare il danno potenziale in caso di rivelazione accidentale della chiave segreta. Il più alto livello possibile di segretezza si ottiene se la chiave segreta è calcolata in più fasi e legata al singolo esemplare di dispositivo. Il setup degli host, così come degli slave, deve però essere effettuato in siti diversi per evitare di compromettere la segretezza del sistema.

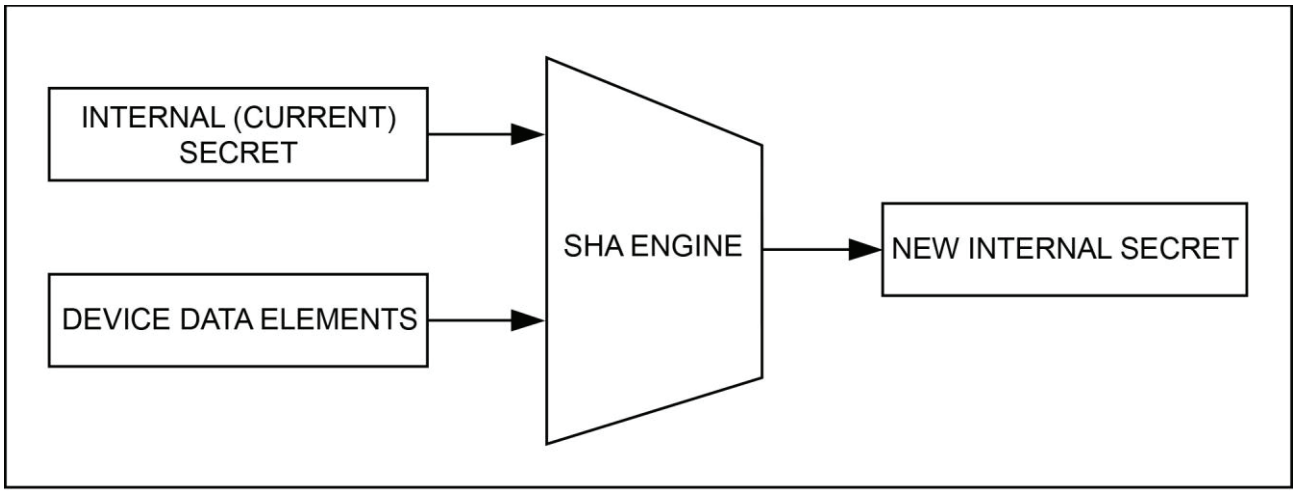


Figura 10. Flusso dei dati per il calcolo di una nuova chiave segreta.

Se incaricato di calcolare una chiave segreta, l’autenticatore sicuro usa il proprio engine SHA-1 o SHA-2 e calcola un MAC usando elementi di dati specifici di quel particolare dispositivo, come illustrato in figura 10. Il MAC risultante viene quindi utilizzato per generare la nuova chiave segreta.

Per Approfondire: http://www.maximintegrated.com/en/products/digital/embedded-security.html